Watermarking for alternative requirements

Teddy Furon

Fundamentals of Digital Image Watermarking

1

Watermarking techniques for alternative requirements

Teddy Furon

1.1 Introduction

So far, watermarking has been presented in this book as a primitive robustly hiding binary messages in host media. It truly reflects the main use of digital watermarking. However, some applications needs adaptations of the basic primitive. The aim of this section is to detail such possible enrichments.

However, this section is not only a list of less classical watermarking applications. The relationship between cryptography and watermarking is the base supporting this chapter. Cryptography and watermarking tackles the same issue: computer security (but note that watermarking doesn't only target secure applications). The presented enrichments are largely inspired by known functionalities of cryptography. The differences are sometimes quite subtle but important. It would not make sense that watermarking only mimics functionalities of cryptography whose theoretical and practical security levels are assessed for decades. Therefore, this chapter focuses on the interactions between these two technologies. All cryptographic references in this chapter are taken from Menezes et al. (1996).

This chapter makes an overview of four different applications: authentication, fingerprinting, watermarking protocols (embedding and detection), and asymmetric watermarking.

1.2 Authentication

Watermarkers call authentication a tool providing the users a means to check the integrity of content. The framework is quite simple: First, a *signer* modifies original

content in a secret way to produce signed content. A second function, the *verifier*, takes a piece of content as an input, and gives a binary output: *true* means that the media is authentic, $false$ means that the media is unsigned or deemed tampered. In this later case, it is desirable to see where the changes have occurred, or to know how the authenticity has be broken.

1.2.1 Cryptographic digital signature and message authentication

This functionality exactly corresponds to a digital signature or a message authentication, except that these cryptographic primitives only work on binary files. This subsection explains in brief how a Digital Signature (DS) and a Message Authentication Code (MAC) work. The book of Menezes et al. (1996) is more than advised.

Let $\mathbf{m} \in \mathcal{M}$ be the message to be signed, and $\mathcal{K}$ the key space. The signing function $S : \mathcal{M} \times \mathcal{K} \to \mathcal{M}$ (i.e., whose domain is $\mathcal{M} \times \mathcal{K}$ and range is $\mathcal{M}$) is typically characterized by a secret key k_S: $\mathbf{m}_S = S(\mathbf{m}, k_S)$. Usually, there is two phases: First, a hash-value of the message is calculated: $\mathbf{h} = \text{Hash}(\mathbf{m})$. Then, it is encrypted with a private key k_S and simply appended to the message: $\mathbf{m}_S = (\mathbf{m}||E(\mathbf{h}, k_S))$. Note that, in this definition, $\mathcal{M}$ is the set of binary messages whose length is finite (not a priori fixed). However, the size of piece of information appended to the message is fixed and not dependent on the original message length.

At the reception of a signed message, the verifier $V : \mathcal{M} \times \mathcal{K} \to \{0, 1\}$ is also a keyed function: $v = V(\mathbf{m}', k_V)$. Usually, there is two phases: It calculates the hash-value of the data: $\mathbf{h}' = \text{Hash}(\mathbf{m}')$. Then, it decrypts the appended code with key k_V and compares both results: $D(E(\mathbf{h}, k_S), k_V) \stackrel{?}{=} \mathbf{h}'$. The equality assesses the integrity of the data.

There is one strong difference between a DS and a MAC. In a MAC scheme, the hash-value is encrypted with a symmetric crypto-system (or, in a more general definition, it is calculated with a secret keyed hash function). This implies that $k_V = k_S$, and that the verifier must be a trusted entity as it can also sign messages. In a DS scheme, the hash is encrypted with an asymmetric crypto-system. This means that the verification key is public and different from the encryption key, which must remain secret. Hence, anyone can check the DS that only the encryption key holder has issued.

Two inseparable key ideas appear:

- *Data integrity.* Data integrity is provided by the extremely 'sensitive' function Hash(.), so that the slightest change in the message yields a completely different hash-value, resulting in an incorrect verification.

- *Data origin authentication.* Only encryption key holders are able to sign the hash. There is only one private key holder in a DS scheme. There are at least two entities sharing the secret key in a MAC scheme (nothing distinguishes the signer from the verifier). A correct verification not only assesses data integrity but also data origin authentication.

In cryptography, it is widely assumed that (or practical applications are such that) the goal of the attacker (if any) is to forge signed messages, not to remove a DS or a MAC. This threat mainly concerns the hash function. It is a mapping from $\mathcal{M}$, the set of messages (with no constraint on the length), to $\mathcal{H}$, the set of L_H bit long hash-values. The domain of definition being larger than the range, several messages share the same hash-value. Cryptographers call them collisions[1]. This issue is tackled by the *one-way* property of the hash function: for a given hash-value, it is computationally impossible to find a suitable message (i.e. a pre-image). This prevents a dishonest user to copy and paste a given DS or MAC to another message, as the probability that its hash-value matches is extremely small for large L_H: 2^{-L_H}. However, the opponent may have a freeway in the choice of the message: for instance, several text messages have the same semantical meaning.

This remark introduces the birthday attack which is the most well-known attack and which could be summed up by: It is easier to accidently find a collision (i.e., two messages having the same hash-value) than to find in purpose a pre-image of a given hash-value. Suppose that the probability mass function of hash-values is uniform, and that the opponent has observed $t = 2^{L_H/2}$ signed messages. Then, he makes slight modifications (i.e., semantically invariant) of the message to be signed without authorization, he calculates the hash-value, and he checks whether it matches one of the observed ones. This collision event is expected after t tries.

Standard signatures' lengths are 2048 bits for RSA (Rivest, Shamir and Adleman), 320 bits for DSA (Digital Signature Algorithm) and around 240 bits for ECDSA (Elliptic Curve based Digital Signature Algorithm). Typical MACs' lengths are 160 bits for SHA-1 (Secure Hash Standard) or, recently recommended 256 bits with SHA-256.

1.2.2 Motivations

The watermarking community has tried to implement the authentication functionality with information hiding tools since 1995 (the first work the author is aware of is Walton (1995), but this activity really started in 1997 with Yeung and F. Mintzer (1997)). Different terms cover the same functionality: fragile watermarking, semi-fragile watermarking, integrity, tamper detection, authentication. This chapter uses the last term provided this covers data integrity and data origin authentication.

Yet, the goal is not to copy the cryptographic primitive. DS and MAC are clearly useful in multimedia protection: what is called a message in the previous subsection can be the binary representation of a piece of multimedia content. For instance, Friedman (1993) proposed to embed in digital camera a secure module which cryptographically signs pictures right before storing them on storage medium. However, some drawbacks motivate the investigation for improvements. Here is a list of the commonly cited arguments:

1. A DS / MAC appended to a message increases its size.
2. There is no clear standard to add a DS / MAC to an image. Hence, this extra information could be lost during a format conversion or an content transmission.

[1] Not to be confused with collusions.

3. If a DS / MAC is incorrect, the user cannot find what is wrong in the image.

4. Simple and common content processings like source coding (e.g., JPEG, JPEG2000) spoil a digital signature, and the encryption key holder is not here to sign again the compressed image. However, users usually apply these transformations without any malicious intention. Similarly, a scalable transmission might drop some low-priority content elements to face variation of channel bandwidth.

The goal of watermarking authentication is to have the key-ideas of cryptographic authentication without these drawbacks. Some basic issues of watermarking authentication are presented in the following subsections.

1.2.3 Watermarking based authentication

Let us first try to get rid off the first two drawbacks (above listed). Denote $\mathcal{X}$ the set of pieces of content of a given size. The signer function is a transformation from $\mathcal{X}$ to $\mathcal{X}$. No increase in content size is thus tolerated. Figure 1.1 sketches the general structure of the signer function: a first process generates the message to be hidden, a second one embeds this message in content thanks to a watermarking technique. These two functions a priori receive a secret key as input. The total key of the signing function is $k_S = \{k_M, k_W\}$, with k_M the secret key for the message generation, and k_W the secret key for the watermarking embedding. Similarly, $k_V = \{k'_M, k_W\}$. Note that the watermarking technique uses the same key at the embedding and at the decoding (see Figure 1.2). The verifier cannot resort to the original image: the watermarking decoder is thus blind. We have to precise the requirements of the watermarking technique: capacity, robustness, imperceptibility. The methods detailed in this subsection are classified into three strategies according to the type of the information to be hidden in content. The first two strategies mimic the MAC scheme as the verifier must be trusted: $k'_M = k_M$. The last one resort to public key encryption as in a DS scheme: $k'_M \neq k_M$.

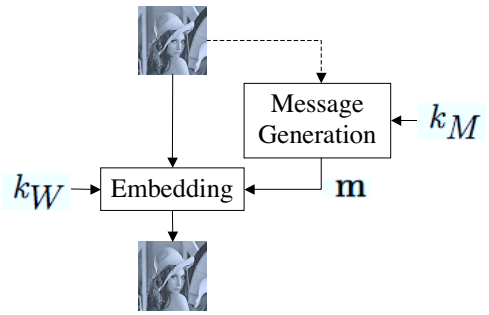

Figure 1.1 Overview of the signer function.

The first (historically) strategy was to hide a piece of information related to the key holder in a so *fragile* way that any (or almost any, see subsection 1.2.5) content

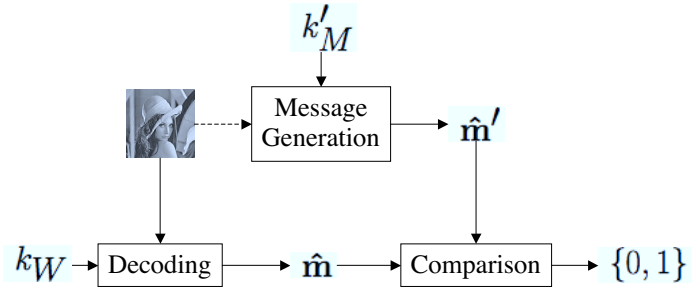

Figure 1.2 Overview of the verifier function.

modification yields a wrong decoded message. We have $\mathbf{m} = f(k_M)$. An opponent cannot forge signed content because he ignores the watermarking key. As watermarking is a symmetric primitive, this idea indeed mimics the MAC functionality. The signer and the verifier share key k_W to embed / decode watermarks and also the knowledge about the piece of information to be embedded: $\mathbf{m} = f(k_M)$. Examples of such schemes are the following articles: Kundur and D. Hatzinakos (1999); Lu and H.-Y. M. Liao (2001); Yeung and F. Mintzer (1997).

Example 1 (The authentication scheme from Yeung and F. Mintzer (1997)) *Denote by* $h(.)$ *a secret hashing function from* $\{0, 1, \ldots, 255\}$ *to* $\{0, 1\}$. *This function applied pixel-wise on 8-bit luminance values of one image yields a binary picture of identical size. Denote by* $\mathbf{m}$ *a* $L_1 \times L_2$ *secret binary logo depending on the secret key* k_M. *The signer slightly changes the pixel values of image* $\mathbf{x}$ *until* $h(x[\mathbf{n}]) = m[n \bmod (L_1, L_2)]$, $\forall \mathbf{n} \in \{0, \ldots, N_1 - 1\} \times \{0, \ldots, N_2 - 1\}$.

The verifier applies the secret function to the luminance value of the pixels, and compares the resulting binary map to the secret logo.

The weakest point of this kind of scheme is that the hidden message does not depend on the content. There is a threat that the opponent simply copies the watermark signal even if he cannot decode it and pastes it within another content. This is known as the copy attack (Kutter et al. (2000)). The watermarking technique cannot be a simple LSB substitution, for instance. An additive scheme is also not recommended as the opponent could average many signed pieces of content to estimate the watermark signal. A quantization scheme is preferable as used by Kundur and D. Hatzinakos (1999). Note that in this case the watermark signal is finally dependent on the host thanks to the side-informed watermarking technique.

This last remark introduces the second strategy, which is not very popular but elegantly based on side information watermarking. Eggers and B. Girod (2001) considers the verifier as a pure watermarking detection process because its output is binary. We insist here on the fact that watermark detection is different than watermark decoding (see chapter 2). The signer modifies the host so that it complies with a given property related to secret key k_W. This is done embedding a fragile watermark signal, which

does not carry any hidden information. This strategy does not follow the structure of Figure 1.1, as there is no message to be hidden. The verifier checks whether received content matches this statistical property. It measures the likelihood that the watermark signal is present in the content. Once again, this looks like the cryptographic MAC primitive as the watermarking detection is not public: $k_S = k_V = k_W$. The probability of false alarm of the detection test must be very low (equivalent of collision prevention). The scheme does not rely at all on any cryptographic primitive (neither hashing nor encryption). Eggers et al. used the well-known side information technique SCS (see chapter 5 for details) so that an average attack is null and vain. Yet, no one did a serious security analysis of their scheme. Note that this looks like the Kundur and D. Hatzinakos (1999) proposal as they both used quantizers. However, Eggers et al. optimize the embedding parameter (α, Δ) with respect to a noise power (see subsection 1.2.5). Moreover, the detection is a soft output (likelihood), whereas D. Kundur used a hard decoding followed by a comparison with the signer's secret message.

The third strategy follows the structure of Figure 1.1. The message to be hidden is indeed a DS / MAC (sometimes called authenticator in watermarking articles). This method is sometimes called *content-based authentication*. Either the hash-value is encrypted with k_M and the watermarking key k_W can be disclosed, either the hash-value is not encrypted and the security stems from the secrecy of k_W. Content modifications either forbid a correct decoding either change the hash-value. The difficulty is that the DS / MAC embedding must not change the hash-value. There are two options: a reversible embedding or the projection of the content onto two orthogonal domains.

Thanks to a reversible watermarking, the decoder outputs the hidden message (i.e., the DS or MAC in our case) and also the original content without any loss. Hence, the verifier can calculate the hash-value of the original content exactly as the signer did. In some applications where classical watermarking is not allowed due to its embedding distortion (e.g. medical imaging), the reversible watermarking is the only solution. The most known example is the scheme from Fridrich et al. (2002).

Example 2 (Reversible watermarking from Fridrich et al. (2002)) *Imagine that a lossless source coding reduces the size of a part* $\mathbf{x}_B$ *of content* $\mathbf{x}$ *(i.e., an image in a given format) to output* $\mathbf{c} = Comp(\mathbf{x}_B)$*. The produced free space* $L_{X_B} - L_C$ *can be used to store the message to be hidden. This results in a reversible data hiding scheme (and more specifically in a watermarking authentication scheme if the message is a DS or a MAC) if the replacement of* $\mathbf{x}_B$ *by* $(\mathbf{c}||\mathbf{m})$ *does not cause perceptible artifacts. The verification process is straightforward. The user removes the hidden message and decompresses* $\mathbf{c}$ *to retrieve the original content. Verification follows the procedure as usual in cryptography. In a classical example,* $\mathbf{x}_B$ *gathers the LSB of pixel luminance values or of quantized DCT coefficients.*

In the second approach, content is projected onto two dual subspaces. Domain $\mathcal{X}_A$ gathers the perceptually important features of the image. It is meaningful to calculate the hash-value from this part, whereas domain $\mathcal{X}_B$ is used to hide the DS / MAC. As the domains are orthogonal subspaces, the embedding in $\mathcal{X}_B$ has no impact on $\mathcal{X}_A$ and thus does not corrupt the hash-value of $\mathbf{x}_B$. Classical choices are MSB / LSB

of pixel luminance values: Celik et al. (2002); Wong (1998); Wong and N. Memon (2001), or low frequencies / high frequencies of spectral coefficients (DCT, Wavelet): Xie and G.R. Arce (2001); Zhao et al. (2004). Zhao et al. (2004) even calculate the hash-value from DC DCT coefficients and insert it in AC Haar transform coefficients. They, of course, proved that these two subspaces are orthogonal.

Example 3 (Public key authentication scheme from Wong (1998)) *Let us split an image (luminance value) into $B_1 \times B_2$ pixel blocks. The image is signed block by block. For a given block, the signer gathers all the the seven first MSB bits of the pixels in a $7B_1B_2$-bit sequence. It calculates a hash-value, which is encrypted it with an asymmetric crypto-system, k_M being the private key. This DS is truncated to B_1B_2 bits and embedded as it is in the LSB of the block pixels. k_W does not exist in this scheme, and $k'_M \neq k_M$ is the public decryption key.*

1.2.4 Local verification

When the verifier states a piece of content is not authentic, the user would like more information, especially about the location of the tampered areas. The verifier would output a map of the non-authentic regions. For instance, this would warns the user whether the image is unsigned (a completely full map), or whether it has been maliciously tampered (some compact areas in the map).

The solution brought by watermarkers is really simple. The schemes so far presented were supposed to work on whole images, but pictures can be split into blocks signed one by one. The smaller the blocks are, the more accurate the localization of tampered areas is. This method is sometimes called block-based authentication. The technique given in example 1 is one the first methods for block-based authentication. It works on pixels, i.e. 1×1 blocks. The most well-studied method is the scheme from Wong and N. Memon (2001), explained in example 3.

However, this approach rises a lot of issues. First, it is obvious that the opponent can replace a block by another signed block without spoiling the authentication. This can be even more dangerous when the forged image is a collage of large parts of pictures signed with the same key. Each part in the composition is separately asserted to be authentic. Boundaries between the merged parts are probably detected as non-authentic. This creates an ambiguity, because one wonders whether only parts of the image have been modified, or if a collage actually took place.

This is the reason why the hash-value of a block usually depends on some extra information such as the size of the image, the position of the block in the image. The only limitation is that the verifier must know these data to calculate the same hash-value. This trick prevents the opponent to replace a block by whatever signed block and it narrows the number of suitable blocks (subsection 1.2.6 develops this potential threat). A similar approach is to calculate a hash-value from data of the block to be signed and also of its neighboring blocks as proposed by Coppersmith et al. (1999). Even this countermeasure has some weaknesses. Two excellent articles analyses and proposes recommendations to safely use this idea Barreto et al. (2002); Deguillaume et al. (2003).

However, this usually decreases the accuracy of the localization of tampered areas to several neighboring blocks. Deguillaume et al. avoid this drawback noticing that a incorrect block signature implies tampered blocks (the actual block and/or its neighborhood), but a valid signature also implies authentic blocks (the actual block and its neighborhood). This rule allows a better localisation of the tampered areas Deguillaume et al. (2003). Celik et al. improve this idea with their hierarchical authenticator Celik et al. (2002).

Example 4 (Hierarchical authentication from Celik et al. (2002)) *Image* $\mathbf{x}_0$ *is decomposed into four children sub-images* $\mathbf{x}_{1,i}$*. This decomposition is iterated* l *times until the smaller sub-images have the size of a elementary block. Image* $\mathbf{x}_0$ *is then composed of* 4^l *blocks* $\mathbf{x}_{l,i}$*. This iterative decomposition can be organized in a pyramidal hierarchical structure. The message to be hidden in image block* $\mathbf{x}_{l,i}$ *is the concatenation of parts of the DS of this block and of all its parents* $\mathbf{x}_{k,j(i,k)}$ *with* $0 \leq k < l$ *(function* $j(i,k)$ *gives the index of the parent block at scale* k *of the* i*-th child block* $\mathbf{x}_{l,i}$*). Figure 1.3 illustrates this message generation. Celik and al. chose the watermarking technique from Wong and N. Memon (2001) (see example 3). At the decoding side, each DS can be reconstituted and checked.*

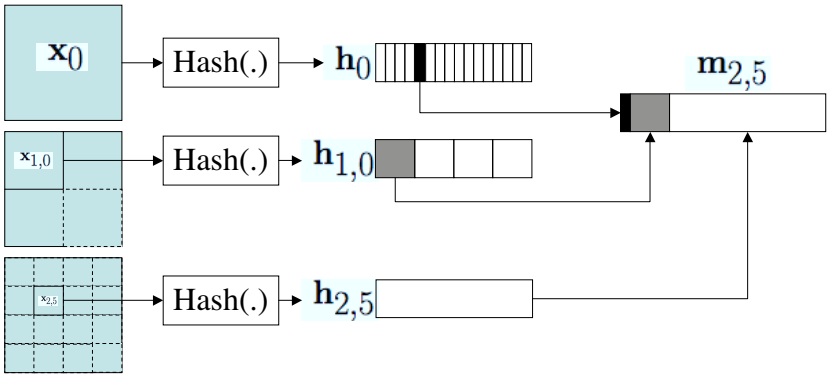

Figure 1.3: Generation of the message to be hidden in image block $\mathbf{x}_{2,5}$. $\mathbf{m}_{2,5}$ is the concatenation of the DS $\mathbf{h}_{2,5}$, with one fourth of DS $\mathbf{h}_{1,0}$, and with one sixteenth of DS $\mathbf{h}_0$.

1.2.5 Semi-fragile watermarking

Watermarkers would like an authentication process decoupled from the binary representation of multimedia content. It is more relevant to assess the integrity of the semantic. It should be possible to authenticate a picture whatever its format. This implies that the signature is robust to a given class of transformations such as format conversions including lossy compressions while, of course, being fragile against the

class of malicious manipulations. Some articles present this idea as *soft* authentication on contrary to *hard* authentication which fades away at any modification (except lossless compression). Others speak about *semi-fragile* versus *fragile* watermarking. Note that fragile watermarking in a transform domain and hard authentication are not compatible. Once the watermark embedded and the inverse transformation done, host samples have to be quantized (usually, 8 bits for pixel luminance, 16 bits for sound sample). The quantization noise might remove the fragile watermark. This is the reason why hard authentication is done in the spatial domain or with the help of an integer-to-integer transform (e.g., the Haar transform in Kundur and D. Hatzinakos (1999)).

Let us focus on the first and second strategies introduced in subsection 1.2.3. A more or less robust watermarking technique allows soft authentication because the message to be hidden is not content dependent in the first strategy and the second strategy does not even need to embed a message. The main issue is the design of a watermarking scheme robust and fragile with respect to two distinct content transformation sets. Usually, there is a trade-off between the perceptibility of the watermark and its robustness. As a quantization based watermarking is widely used in authentication, this trade-off is achieved selecting heuristically an appropriate quantization step. Here are some examples of embedding domains (i.e., the coefficients extracted from the original image which support the quantization process): median values of some selected wavelets coefficients (Xie and G.R. Arce (2001)), wavelet coefficients (Kundur and D. Hatzinakos (1999); Lu and H.-Y. M. Liao (2001)), sums of Haar transform coefficients (Zhao et al. (2004)), or DCT coefficients (Eggers and B. Girod (2001)). For instance, in the later reference, the quantization step (and also the scaling factor α, see chapter 5) is tuned so that the watermarked image has a PSNR higher than 40dB while being robust to a JPEG compression down to a PSNR of 36dB.

Another possibility is to let the user decide whether the content has suffered from a malicious or non-malicious modification. The verifier calculates the likelihood of a modification event like the approach of Eggers and B. Girod (2001). It is expected that this score is null if the content is authentic (hard authentication), small if some light (hence pretendingly non-malicious) content transformations has been performed (soft authentication), and high if the opponent has modified this piece of content (forgery) or if it has not been signed. It is up to the user to set the threshold between soft authentication and forgery. Some statistical considerations can help him. For instance, in Kundur and D. Hatzinakos (1999), the verifier calculates the bit-error rate (the decoder knows the host independent message, i.e. it follows the first strategy). This measure equals zero in case of successful hard authentication and, in expectation, 1/2 in case of unsigned content. A threshold and the probabilities of a miss and of a false alarm (respectively, the probability that a softly distorted content is deemed tampered, and the probability that a tampered content is considered as soft authentic) is provided with the assumption that a soft content transformation behaves like the addition of white gaussian noise on the quantized coefficients. Calculus looks like those shown in chapter 2, Eq.(2.231) and Eq.(2.232). In the same way, Zhu et al. (2004) estimate the mean square error between the observed coefficients and their expected values (knowing the bits they should carry).

Example 5 (Distortion measurement technique from Zhu et al. (2004)) *The signer modifies the* 8×8 *DCT coefficients* $\mathbf{x}$ *of the image as follows:*

$$y[\mathbf{n}] = M_f[\mathbf{n}] \left(\left\lfloor \frac{x[\mathbf{n}]}{M_f[\mathbf{n}]} \right\rfloor + r[\mathbf{n}].sign(x[\mathbf{n}]) \right), \tag{1.1}$$

where $\mathbf{r}$ *is a secret random sequence of values in* $[0, 1)$*,* $M_f[\mathbf{n}]$ *a JND masking value for the considered DCT coefficient. The inverse DCT gives the signed picture.*

The verifier estimates the distortion for each DCT coefficient:

$$\epsilon[\mathbf{n}] = v[\mathbf{n}] - M_f[\mathbf{n}] \left(\left\lfloor \frac{v[\mathbf{n}]}{M_f[\mathbf{n}]} - (r[\mathbf{n}] - 0.5).sign(v[\mathbf{n}]) \right\rfloor + r[\mathbf{n}].sign(v[\mathbf{n}]) \right). \tag{1.2}$$

Let us now deal with the third strategy of subsection 1.2.3. Soft authentication implies a semi-fragile watermarking but also a soft hash function because a mild transformation must not corrupt the hash-value. This resorts to a recent activity of multimedia research known as *soft hash function* or *passive fingerprinting*, whose field of application is larger than the sole authentication purpose. In Wu (2002), the hash-value is a sequence of the LSB of some selected quantized features (such as DCT coefficients). The author shows for his scheme that there exist a maximal distortion D_A that a signed image can bear until its hash-value changes. In Lin and Chang (2001), the hash-value is based on comparison between coefficients at the same frequency bin in separate 8×8 DCT blocks of an image[2]. These relationships are invariant by JPEG compression as DCT coefficients of a given frequency bin are quantized with the same step. In the same way, Zhao et al. (2004) compares DC DCT coefficients to generate the hash-value. In Xie and G.R. Arce (2001), the hash-value is the binary map issued by an edge detector on a down-sampled version of the image (for instance, a Sobel detector on the LL subband of a three level wavelet decomposition).

All these examples show that the heuristic approach so far prevails in the young research area of soft hash function.

1.2.6 Attacks on authentication schemes

The watermarking schemes studied in this section are pretendingly offering the same functionality as the authentication cryptographic primitive. Therefore, a security analysis is mandatory when introducing such new primitives. Yet, very few articles deal with this issue. A lot of them only check that if the opponent replace parts of the image by independent pieces of content (e.g., removal or pasting of object in picture), then the probability that the hash-value is still valid is extremely small. However, we have already noted that block-wise authentication might lead to security flaws. This argues that a more proper analysis of what the pirate can do is of utmost importance.

The starting point of the security analysis is the assumption that the opponent is smart, that he knows the algorithms of the authentication scheme (Kerckhoffs

[2]However, Lin and Chang (2001) and Wu (2002) propose improvements of the Friedman (1993) scheme, i.e. the hash-value is encrypted and appended beside content as metadata. It is not embedded in the image.

principle), and that he has access to many signed pieces of content (J.Fridrich (2002) has considered other types of attacks). The only thing the opponent is missing is the secret key of the signer.

In the first and second strategies of subsection 1.2.3, the message to be hidden and the watermarking key are secret. The scheme seems secure if the watermark copy attack is not possible and if one proves that the key remain secret while the opponent observes more and more pieces of content signed with the same key. However, when local verification is enable, these strategies are weak against blocks replacement.

In the third strategy of subsection 1.2.3, the signing process is in two steps: the generation of the message to be hidden and its embedding. Assume for the moment that both of them are private (i.e., keyed by k_W and k_M). A first issue is to be sure that the exchange of two signed blocks is detectable by the verifier as suggested in subsection 1.2.4. Here are the classical counter-measures:

- The hash-values of the image-blocks also depend of the size of the whole image. This prevents pirates to crop the signed image. Moreover, blocks replacement only occurs between images of the same dimensions.
- The hash-values of the image-blocks also depend on the location of the block in the whole image. Thus, the replacement attack is limited to blocks located in the same location in images.
- The hash-values of the image-blocks also depend on the image ID. The replacement attack is limited to blocks of the same image. This counter-measures is hardly practical as the verifier must knows the image ID as well. This ID can be embedded in the picture with a robust watermark technique Deguillaume et al. (2003).

A mix of these counter-measures dramatically reduces the size of the set of replacement blocks.

Suppose, the scheme mimics a MAC with a keyed-hash (it is not a public hash function followed by an encryption). The opponent doesn't know anything as far as the watermarking key remains secret.

Suppose the scheme is a MAC with a public hash function or a DS. Then the adversary knows the hash-values of the blocks or their embedded messages. He notices whether two blocks share the same hash-value (MAC) or the same signature (DS). Note that it is not a big deal if the hash-value depends on some extra data (position of the block, image ID, data of neighboring blocks) provided that the attacker knows them to calculate hash-values. The opponent is able to classify the blocks in classes of equivalence (also called a codebook): two blocks are equivalent if they share the same hash-value (MAC) or if they share the same hidden messages (DS). For DS schemes, the verification is public so that this classification is always possible.

There are many different versions of this hack based on classification, depending on the context. A first distinction is whether the watermarking key is disclosed (or public). If it is not, like in MAC schemes, the attacker is not able to watermark his own pieces of content, but he can exchange two signed blocks. This version of the birthday attack is known as the Vector Quantization attack or the Holliman-Memon

attack Holliman and N. Memon (2000). The attacker has to find within the codebook of blocks whose hash-value is correct (knowing the extra data), the codeword which is the closest to the desired replacement block for some distortion measure.

If the watermarking key is public (DS schemes), the opponent can embed messages. Yet, he cannot generate messages as key k_M is still missing. A Vector Quantization attack works, but another strategy is possible: the birthday attack. He slightly distorts his desired replacement block in a way that its semantic meaning remains the same, until the hash-value of the block (knowing the extra data) matches one of the observed hash-values of signed blocks. Finally, he embeds in the modified desired block the corresponding hidden message. He has thus succeeded to sign a block while still ignoring key k_M.

Note that a birthday attack is in a way always possible in DS schemes, and that the only counter-measure is to use long hash-values[3]. The important thing is that there shall not be not any attack less complex than the birthday attack. From this point of view, the first version of the scheme from Wong (1998) was really unsecure as the opponent could exchange whatever signed blocks. Denote L_L the number of blocks in an image, and suppose that the L_I observed signed images have the same size. The opponent can lead a VQ attack with only one global 'codebook' containing $L_I L_L$ blocks. With the dependence on the block location (see the above-mentioned counter-measures), there are now L_L separated codebooks. The attack is harder as each codebook only contains L_I image blocks. The probability to find, in a given codebook, an image block perceptually close to the one the opponent is willing to paste in is much smaller. With the dependence on the block location and the image ID, there are now 2^{L_H} codebooks, which is the greatest figure of codebooks, and which renders VQ attack the least probable as each of them only contains on average $L_I L_L 2^{-L_H}$ blocks. Finally, the hierarchical authentication scheme from Celik renders the birthday attack almost impossible. A modification in an elementary block changes the hash-values of the block and of its parents. Hence, the opponent has to carry $L_B = 1 + 4 + \ldots + 4^l = (4^{l+1} - 1)/3$ birthdays attacks in parallel, which decreases the probability of success with an exponential rate of L_B. However, its complete security assessment requires more development.

Another point which lacks security analysis is the hash generation in soft authentication. The birthday attack is considered from a probabilistic point of view in cryptography due to the one-way property of the hash function. The opponent cannot generate pre-image of hash-values, hence, he changes his message until its hash-value matches one of the observed signatures. The hash-value is then regarded as an uniformly distributed random variable. With the soft hash functions of subsection 1.2.5, the one-way property is absolutely not guaranteed. Hence, it might be far easier to create collisions. As far as the author knows, this matter has never been studied.

Finally, there is a compromise between the security level and the localization of tampered areas. Too small blocks won't allow the embedding of long length messages. The problem is even bigger when semi-fragile embedding is envisaged as their capacity is often smaller than the one of fragile embedding. Typical DS/MAC lengths given in subsection 1.2.1 are indeed quite big in comparison with usual watermarking capacity.

[3]There exist counter-measures in cryptography but they are out of scope in this tutorial chapter.

1.2.7 Conclusion about authentication

Authentication of multimedia content is very interested as it questions the important role of images and recorded sound in our society. Images are extremely powerful media but can we trust them? We answer that image authentication by watermarking becomes to be mature to spot image modifications made with classical photo editing software. However, it is not bringing security levels high as those of cryptographic digital signature or message authentication. Hence, watermarking for authentication is not yet ready to face scenarii where clever pirates have huge benefits manipulating images.

Keeping this security point of view, we think that the most significant references in authentication are the following ones: Deguillaume et al. (2003),Barreto et al. (2002) and J.Fridrich (2002).

1.3 Fingerprinting

Fingerprinting is a technique to embed users' identifier $\hat{\mathbf{m}}_i$ in original content $\mathbf{x}$ producing a protected content $\mathbf{y}_i$, in order to trace back this information in pirated content. When a piece of content $\mathbf{v}$ is caught in unauthorized hands, the identity of the 'traitor' is disclosed by the decoded marks $\hat{\mathbf{m}}$. Fingerprinting has a long history. In the past, manufacturers introduced tiny errors or imperfections in maps, diamonds or logarithm tables to hide the name of a recipient. This chapter only deals with fingerprinting nowadays renaissance in the context of multimedia content protection.

1.3.1 Link with cryptography

Fingerprinting is related to cryptographic primitives dealing with key management in communication groups. The rationale underlying the application is the following.

Imagine that dishonest users cannot give decrypted content to unauthorized people due to the huge amount of data compared to the available bandwidth. They prefer to forge pirated control access keys. If the key management deals the same key to all users, then it is not possible to trace back the traitor who discloses his key. If the key management attributes a key per member, it is extremely easy to find the traitor. However, this requires transmitted data to encrypted in as many versions as there exist decryption keys. This completely spoils the transmission rate. Chor et al. proposed broadcast encryption system where data are encrypted in reasonable amount of times and users have a set of decryption keys Chor and M. Naor (1994) which allows them to decrypt data at a high probability of success. In the other hand, users do not share the same keys. This diversity allows to trace back dishonest users even if they have shared their keys to build a pirated device.

Now, if the amount of data is not so big, then dishonest users change their strategy. It is easier to decrypt the data (as any user can do it) and to share content in the clear with unauthorized people. This is the reason why the embedding of users ID into content might dissuade the pirates. This embedded message $\mathbf{m}_i$ is unique for each member and it plays the role of the client's fingerprints let on possessed object. As far as the author knows, fingerprinting is not a receivable legal proof to sue the traitor

in court as it does not prove the intentionality of the act. However, in the framework of a pay service, the protection system might threat the traitors of canceling their membership (some say that dishonest users are blacklisted).

Nevertheless, several dishonest users might gather their knowledge of the system and the pieces of content $\mathbf{y}_i$ in order to forge pirated content $\mathbf{v}$. This group of people is named a *collusion*[4]. Figure 1.4 illustrates the game between the protection system which tries to trace the traitors, and these dishonest users. Of course, the probability of accusing an honest user must be extremely small.

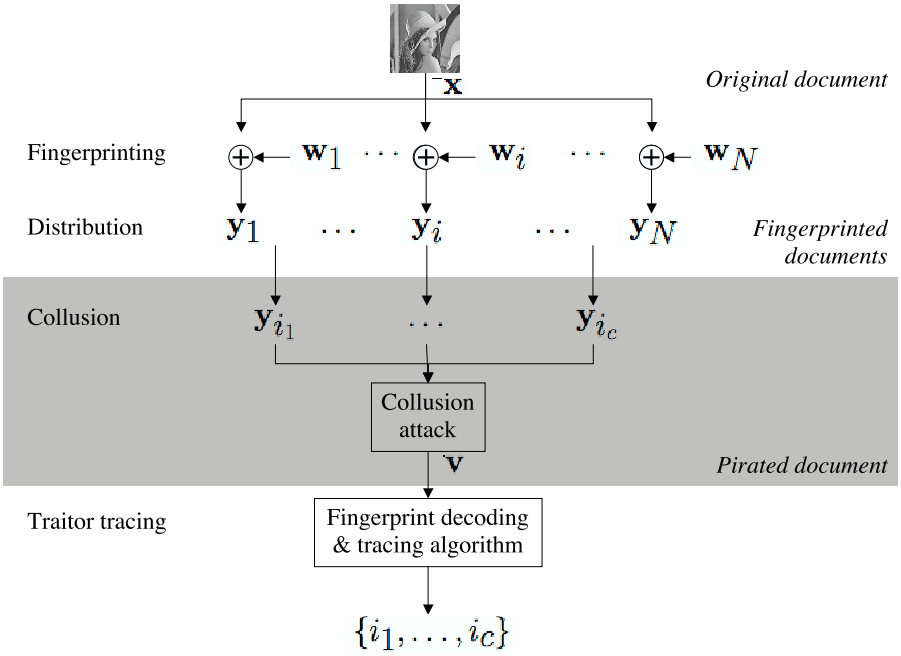

Figure 1.4 Overview of the fingerprinting framework.

1.3.2 Outlines of the study

Fingerprinted is not only linked with cryptography as mentioned above, but it is mostly studied by cryptographers. The concept was introduced in Wagner (1983), and it has gained interests since Boneh and Shaw work Boneh and J. Shaw (1998). The fingerprinting problem is decomposed into the following issues: setup of a mathematical model of the collusion attack, definition of features of the code (set of identifiers), construction of such a code. In a way, the assumption is that watermarkers will embed

[4]This term is used with a lot of confusion in the watermarking community. For instance, it is often used instead of an average attack. Remember that collusion necessary means a group of people, not an image processing.

fingerprinting codes developed by cryptographers. Thanks to the mathematical model of the collusion attack, the conception of codes is decoupled from the physical layer (ie., the watermarking channel). The reader must understand that this is not assessed theoretically, but it is an historical framework which will prevail in our overview. The first parts (section 1.3.4 to 1.3.7) reflect the large literature available in cryptography, whereas the last part questions the embedding of such codes in digital content.

This study does not present some rare fingerprinting schemes with additional properties like asymmetric fingerprinting (only users know their fingerprinted version of content Pfitzmann and Schunter (1996)), anonymous fingerprinting (users are anonymous, until the tracing algorithm proves the registration center a particular client is dishonest Pfitzmann and Waidner (1997)), dynamic or sequential traitor tracing (tracing is done sequentially where a caught traitor is blacklisted at each step Safavi-Naini and Wang (2003)).

1.3.3 Notations

Let $\mathcal{Q}$ be an alphabet of q symbols and $\mathcal{Q}^L$ a vector space. A (L, N, q)-code $\mathcal{C}$ of length L and size N on alphabet $\mathcal{Q}$ is a set of N sequences of L symbols $\mathbf{m}_i = (m_i[1], \ldots, m_i[L])$, $m_i[j] \in \mathcal{Q}\ \forall i \in \{1, \ldots, N\}, \forall j \in \{1, \ldots, L\}$. A word is an element of $\mathcal{Q}^L$, named codeword if it belongs to $\mathcal{C} \subseteq \mathcal{Q}^L$. A (L, N, q)-code $\mathcal{C}$ is also represented by a $N \times L$ matrix whose entries are in $\mathcal{Q}$. $\forall \mathbf{m}, \mathbf{n} \in \mathcal{Q}^L$, define the set $\mathcal{I}(\mathbf{m}, \mathbf{n}) = \{j | m[j] = n[j]\}$. Extend this definition onto subsets of $\mathcal{Q}^L$: $\mathcal{I}(\{\mathbf{m}_1, \ldots, \mathbf{m}_c\}) = \{j | m_1[j] = \ldots = m_c[j]\}$. Define the Hamming distance between $\mathbf{m}, \mathbf{n} \in \mathcal{Q}^L$ by $d(\mathbf{m}, \mathbf{n}) = L - |\mathcal{I}(\mathbf{m}, \mathbf{n})|$, and the distance between $\mathbf{m} \in \mathcal{Q}^L$ and a set $\mathcal{S} \subseteq \mathcal{Q}^L$ by $d(\mathbf{m}, \mathcal{S}) = \min_{\mathbf{n} \in \mathcal{S}} d(\mathbf{m}, \mathbf{n})$.

A collusion $\mathcal{C}_o$ of c dishonest users is defined by the set of the c codewords embedded in their fingerprinted version of the content: $\mathcal{C}_o = \{\mathbf{m}_{i_1}, \ldots, \mathbf{m}_{i_c}\}$. The collusion attack yields a pirated content $\mathbf{v}$, whose decoding gives a word $\hat{\mathbf{m}}$. The set of all possible words output by this collusion is denoted $\text{desc}(\mathcal{C}_o)$ and called the set of descendants of $\mathcal{C}_o$ in Staddon et al. (2001) (or envelope in Barg et al. (2003), or feasible set in Boneh and J. Shaw (1998)). This set is strongly dependent on the assumptions about the embedding and the collusion attack processes. The concept of pirated copy and traitors are now represented by the descendant and its parents respectively. Given c a positive integer, we define the c-descendant code of $\mathcal{C}$ as: $\text{desc}_c(\mathcal{C}) = \bigcup_{\mathcal{C}_o \subset \mathcal{C}, |\mathcal{C}_o| \leq c} \text{desc}(\mathcal{C}_o)$. The descendant $\hat{\mathbf{m}}$ belongs to $\text{desc}_c(\mathcal{C})$, the goal of the tracing algorithm is to find back the parents $\mathcal{C}_o$.

1.3.4 The marking assumption

The marking assumption is a terminology coming from Boneh and J. Shaw (1998). A mark is a position in the content which can be in one of q different states, without causing perceptual artefact (or without causing the content useless). Thus, one can hide a symbol of $\mathcal{Q}$ in a mark. We suppose that there are at least L marks in the content.

In most articles, the collusion attack consists in building the pirated content $\mathbf{v}$ by randomly selecting blocks among the available fingerprinted versions $\mathbf{y}_i$ of the

collusion. This gives the following set of descendants:

$$\text{desc}(\mathcal{C}_o) = \{\mathbf{m} | m[j] \in \{m_{i_1}[j], \ldots, m_{i_c}[j]\}\}. \quad (1.3)$$

Denote this marking assumption $A1$. It is used in Fernandez and Soriano (2004b); Staddon et al. (2001), and called narrow-sense envelope in Barg et al. (2003). $A1$ is assumed by default in this chapter unless mentioned exception.

Narrow or wide sense fingerprinting problem

However, the dishonest users of a collusion can spot the marks where the hidden symbols differ, and modify in some way these hidden symbols. The undetected marks are unchanged, and their indices are given by $\mathcal{I}(\mathcal{C}_o)$. This is the marking assumption as stated in Boneh and J. Shaw (1998).

This stems in a first variation presented in Barg et al. (2003). In the narrow sense problem, colluders replace a detected mark by randomly selecting one of their marks at that position, whereas in the wide-sense problem, colluders replace a detected mark by randomly selecting a symbol in the alphabet. Thus, in the wide sense problem, the descendant set is defined by:

$$\text{Desc}(\mathcal{C}_o) = \{\mathbf{m} \in \mathcal{Q}^N | m[j] = m_{i_1}[j] \text{ for } j \in \mathcal{I}(\mathcal{C}_o)\}. \quad (1.4)$$

We use a capital D to denote the descendant set in the wide sense problem. Denote this assumption $A2$. Note that this distinction does not hold in the binary case.

Erasures

The second variation is the presence of erasures. Boneh and Shaw introduce the case where the colluders can erase the detected marks Boneh and J. Shaw (1998). Erasures are denoted by '$*$'. The descendant set is denoted desc* in this case:

$$\text{desc}^*(\mathcal{C}_o) = \{\mathbf{m} \in (\mathcal{Q} \cup \{*\})^L | m[j] = m_{i_1}[j] \text{ for } j \in \mathcal{I}(\mathcal{C}_o)\}. \quad (1.5)$$

Denote this marking assumption $A3$.

This has been generalised by Guth and Pfitzmann in a *weak marking assumption*, where a bounded number of erasures might occur everywhere in the sequence:

$$\text{desc}^{**}(\mathcal{C}_o) = \{\mathbf{m} | m[j] \in \{m_{i_1}[j], \ldots, m_{i_c}[j]\} \cup \{*\} \text{ and } |\{j : m[j] = *\}| \leq e\}. \quad (1.6)$$

Denote this marking assumption $A4$. It is used, for instance, in Safavi-Naini and Wang (2001). We can also defined Desc$^*(\mathcal{C}_o)$ (Barg et al. (2003)) and Desc$^{**}(\mathcal{C}_o)$.

1.3.5 Types of codes

Let $\mathcal{C}$ be an (L, N, q)-code and $c \geq 2$ an integer. A perusal of the fingerprinting literature in cryptography shows a large diversity of types of codes. Thanks to the overview of J. Staddon Staddon et al. (2001), we have the following classification.

Frameproof

$\mathcal{C}$ is c-frameproof (c-FP) if

$$\forall \hat{\mathbf{m}} \in \text{desc}_c(\mathcal{C}), \quad \hat{\mathbf{m}} \in \text{desc}(\mathcal{C}_o) \cap \mathcal{C} \rightarrow \hat{\mathbf{m}} \in \mathcal{C}_o. \tag{1.7}$$

This can also be stated as

$$\forall \mathcal{C}_o \subset \mathcal{C} : |\mathcal{C}_o| \leq c, \quad \text{desc}(\mathcal{C}_o) \cap \mathcal{C} = \mathcal{C}_o. \tag{1.8}$$

It means that no collusion can frame another user not in the collusion by producing its codeword.

This feature is essential in fingerprinting. Imagine the scenario where the system expects no collusion: the dishonest user redistributes his copy without any change. If we are naive, we believe that this problem doesn't require a complex code: We randomly associate a word of $\mathcal{Q}^L$ to each different user, with $L = \log_q N$. It means that $\mathcal{C} = \mathcal{Q}^L$. The system observes a pirated content with fingerprint $\hat{\mathbf{m}}$, and it naturally accuses the associated user. However, this system doesn't work as the caught user could argue that a collusion has framed him. He pretends he is not guilty but the victim of a collusion. Because $\mathcal{C} = \mathcal{Q}^L$ is indeed not frameproof, his argument holds. This illustrates the need of a frameproof code.

Example 6 (Optimal N-frameproof code from Boneh and J. Shaw (1998)) *$\Gamma^{(6)}$ is a simple $(N, N, 2)$-code, defined over the binary alphabet $\mathcal{Q} = \{0, 1\}$. It is composed of the codewords $\mathbf{m}_i$ such that $m_i[j] = 0, \forall j \neq i$, and thus $m_i[i] = 1$. $\Gamma^{(6)}$ is N-frameproof. Boneh and Shaw prove that any $(L, N, 2)$-code which is N-frameproof, has a length $L = N$. Thus, $\Gamma^{(6)}$ is optimal in this sense.*

Secure frameproof

A c-secure frameproof code is also called a (c, c)-separating code, for instance, in Barg et al. (2003). $\mathcal{C}$ is c-secure frameproof (c-SFP) if $\forall \hat{\mathbf{m}} \in \text{desc}_c(\mathcal{C})$,

$$\hat{\mathbf{m}} \in \text{desc}(\mathcal{C}_o) \cap \text{desc}(\mathcal{C}'_o) \rightarrow \mathcal{C}_o \cap \mathcal{C}'_o \neq \emptyset. \tag{1.9}$$

It means that no collusion $\mathcal{C}_o$ can frame a disjoint collusion $\mathcal{C}'_o$ by producing a descendant of $\mathcal{C}'_o$. As in the above subsection, a dishonest user cannot pretend he is a victim of a collusion. If $\hat{\mathbf{m}} \in \text{desc}(\mathcal{C}_o) \cap \text{desc}(\mathcal{C}'_o)$, then the tracing algorithm accuses a user in $\mathcal{C}_o \cap \mathcal{C}'_o$. However, this does not guarantee the success of decoding! For instance, imagine $\hat{\mathbf{m}} \in \text{desc}(\mathcal{C}_o) \cap \text{desc}(\mathcal{C}'_o) \cap \text{desc}(\mathcal{C}''_o)$, but $\mathcal{C}_o \cap \mathcal{C}'_o \cap \mathcal{C}''_o = \emptyset$, it is not possible to accuse any user without taking the risk of accusing an innocent. This introduces two branches of fingerprinting codes: either, we strengthen the feature of the code in order that such an event is impossible (see subsections below and the section 1.3.6 about strong traceability), either the probability of this event, called error probability, is small and decreasing as the code length increases Barg et al. (2003); Boneh and J. Shaw (1998). This stems in the concept of probabilistic fingerprinting, or weak traceability, which is studied in section 1.3.7.

Example 7 *Code* $\Gamma^{(6)}$ *presented in example 6 is not* 2*-SFP. For instance, for* $N = 4$*, the word* $(0,0,0,0)$ *can be produced by the collusion of codewords* $(1,0,0,0)$ *and* $(0,1,0,0)$*, or the collusion of codewords* $(1,0,0,0)$ *and* $(0,0,1,0)$ *(or any collusion of 2 codewords indeed).*

Example 8 (Boneh and Shaw elementary code Boneh and J. Shaw (1998)) *The following code* $\Gamma^{(8)}$ *is a* $(3,4,2)$*-code which is* 2*-SFP:*

$$\begin{aligned} \mathbf{m}_1 &= (1,1,1) \\ \mathbf{m}_2 &= (1,0,0) \\ \mathbf{m}_3 &= (0,1,0) \\ \mathbf{m}_4 &= (0,0,1) \end{aligned}$$

The collusion $\{\mathbf{m}_1, \mathbf{m}_2\}$ *gives any word like* $\hat{\mathbf{m}} = (1,?,?)$*, whereas the collusion* $\{\mathbf{m}_3, \mathbf{m}_4\}$ *gives* $\hat{\mathbf{m}} = (0,?,?)$*. In the same way, the collusion* $\{\mathbf{m}_1, \mathbf{m}_3\}$ *gives any word like* $\hat{\mathbf{m}} = (?,1,?)$*, whereas the collusion* $\{\mathbf{m}_2, \mathbf{m}_4\}$ *gives* $\hat{\mathbf{m}} = (?,0,?)$*. Last, the collusion* $\{\mathbf{m}_1, \mathbf{m}_4\}$ *gives any word like* $\hat{\mathbf{m}} = (?,?,1)$*, whereas the collusion* $\{\mathbf{m}_2, \mathbf{m}_3\}$ *gives* $\hat{\mathbf{m}} = (?,?,0)$*. Hence, two disjoint collusions cannot produce the same word. Note that* $\Gamma^{(8)}$ *is also* 2*-frameproof.*

Identifiable parent property

$\mathcal{C}$ has the identifiable parent property (c-IPP) if $\forall \hat{\mathbf{m}} \in \mathrm{desc}_c(\mathcal{C})$,

$$\bigcap_{\mathcal{C}_o | \hat{\mathbf{m}} \in \mathrm{desc}(\mathcal{C}_o)} \mathcal{C}_o \neq \emptyset. \tag{1.10}$$

It means that no collusion can produce a word that cannot be traced back to at least one member of the collusion. The goal here is to avoid the case where the potential collusions are indeed disjoint sets as mentioned above. This gives birth to strong traceability (see section 1.3.6).

Example 9 *Code* $\Gamma^{(8)}$ *is not* 2*-IPP. The word* $\hat{\mathbf{m}} = (0,0,0)$ *for instance can be produced by collusions* $\{\mathbf{m}_3, \mathbf{m}_4\}$*,* $\{\mathbf{m}_2, \mathbf{m}_4\}$*, or* $\{\mathbf{m}_2, \mathbf{m}_3\}$*. The system is not able to reliably accuse one traitor. More precisely, it can accuse one of these three users but with a probability of error of* $1/3$*.*

Example 10 (First position code from Staddon et al. (2001)) *Let* $\mathcal{Q} = \{1,2,3,4\}$*.* $\Gamma^{(10)} = \{(1,2,2); (2,3,4); (3,2,2); (4,4,3)\}$ *is a* $(3,4,4)$*-code which is* 4*-IPP. Note that the 4 codewords have a different symbol in the first position. Thus,* $w[1]$ *indicates which user is surely in the collusion. The intersection of all possible collusions (ie., such that* $\hat{\mathbf{m}} \in desc(\mathcal{C}_o)$*) contains at least this user.*

Traceability

$\mathcal{C}$ is a c-traceability code (c-TA), if $\forall \hat{\mathbf{m}} \in \mathrm{desc}(\mathcal{C}_o)$

$$\exists \mathbf{n} \in \mathcal{C}_o \, : \, \forall \mathbf{m}_i \in \mathcal{C} \backslash \mathcal{C}_o, |\mathcal{I}(\hat{\mathbf{m}}, \mathbf{n})| > |\mathcal{I}(\hat{\mathbf{m}}, \mathbf{m}_i)|. \tag{1.11}$$

It means that the pirated word $\hat{\mathbf{m}}$ is closer to $\mathcal{C}_o$ than to $\mathcal{C}\backslash\mathcal{C}_o$. Tracing traitors reduces to searching for the codewords that agree in most symbol positions with $\hat{\mathbf{m}}$. In short, $\mathcal{C}$ is a c-TA code if $d(\hat{\mathbf{m}}, \mathcal{C}_o) < d(\hat{\mathbf{m}}, \mathcal{C}\backslash\mathcal{C}_o)$.

The tracing algorithm cannot expect to find all parents, since some of them may contribute with too few positions and cannot be traced. Any codeword that is involved in the construction of the descendant in an unambiguous way is defined as a positive parent.

Example 11 *Code* $\Gamma^{(10)}$ *is not* 2*-TA. For instance, the word* $\hat{\mathbf{m}} = (2, 2, 2)$ *is forged by the collusion* $\{\mathbf{m}_1, \mathbf{m}_2\}$*. Let us calculate the distances:*

$$\begin{aligned} d(\hat{\mathbf{m}}, \mathbf{m}_1) &= 1, \\ d(\hat{\mathbf{m}}, \mathbf{m}_2) &= 2, \\ d(\hat{\mathbf{m}}, \mathbf{m}_3) &= 1, \\ d(\hat{\mathbf{m}}, \mathbf{m}_4) &= 3. \end{aligned}$$

The innocent user 3 is as closed to $\hat{\mathbf{m}}$ *than guilty user 1 is. Thus, this code is not* 2*-TA. This is the reason why the tracing of* $\hat{\mathbf{m}}$ *by looking for the* $c = 2$ *nearest neighbors is absolutely not recommended if* $\mathcal{C}$ *is not TA: Guilty user 2 is not suspected whereas innocent user 3 is accused.*

Example 12 (Trivial N-TA code from Staddon et al. (2001)) *The following* (L, N, N)*-code* $\Gamma^{(12)}$ *is* N*-TA:* $\Gamma = \{(1, \ldots, 1); \ldots; (q, \ldots, q)\}$. $\forall \mathbf{m}_i \in \mathcal{C}\backslash\mathcal{C}_o$*, symbol 'i' never appears in* $\hat{\mathbf{m}}$*. Thus* $d(\hat{\mathbf{m}}, \mathbf{m}_i) = L$*. The value of* $\hat{w}[1]$ *proves user '*$w[1]$*' to be part of the collusion. We have* $d(\hat{\mathbf{m}}, \mathbf{m}_{w[1]}) \leq L - 1$*. This shows this trivial code is* N*-TA.*

Conclusion

It is time to establish relationship between the types of code as already suggested by the suite of examples.

Theorem 1.3.1 (Relationship between types of code Staddon et al. (2001)) *Under A*1*, we have the following structure between the types of code:*

- *c-TA implies c-IPP,*
- *c-IPP implies c-SFP,*
- *c-SFP implies c-FP.*

In general, the converse is not true.

The following point is extremely important. We want at least to assess that a collusion cannot frame an innocent user (c-FP) or an innocent group of users (c-SFP). This basic property is also enforced by c-IPP and c-TA. But, these two last types of codes add another property: the success of identifying at least one colluder. This property is only fulfilled in a probabilistic way by c-FP and c-SFP (a probability of error to be bounded). Hence, we have to split the study into two parts: Strong traceability (c-IPP and c-TA) versus weak traceability (c-FP and c-SFP).

1.3.6 Strong traceability

Strong traceability is a very hard requirement and typical code lengths are extremely huge. We start the study mentioning some bounds known in neighboring mathematical concepts such as cover-free families and hash families. These bounds do not give any clue to construct such codes. The second subsection lists some tools useful to build and decode traceability codes.

Theorems

Suppose that q, L, and c are fixed. We want N as large as possible. Here are some bounds on the tractable number of users.

Theorem 1.3.2 (Bound from separating hash families Staddon et al. (2001)) *In a (L, N, q) c-TA code (or c-IPP, or c-SFP), $N \leq q^{\lceil L/c \rceil} + 2c - 2$.*

This bound is the best known for $c > 2$. For 2-IPP codes, a stronger bound is the following

Theorem 1.3.3 (Bound for 2-IPP codes Hollmann et al. (1998)) *In a (L, N, q) 2-IPP code, $N \leq 3q^{\lceil L/3 \rceil}$.*

Tools for codes construction

Error correcting codes Before exploring relationship between Error Correcting Codes (ECC) and strong traceability, here is a short summary on ECC. A code $\mathcal{C}$ is a linear code if it forms a subspace of $\mathcal{Q}^L$. For a linear (L, N, q)-ECC with dimension $k = \log_q N$ and minimum distance d, the inequality $d \leq L - k + 1$ always holds (Singleton bound). ECC with equality are called maximum distance separable codes (MDS codes). A well-known class of linear MDS codes are Reed-Solomon codes ($L = q - 1$, $k = q - d$). For a (L, N, q)-ECC with minimum distance $d > (1 - 1/q)L$, then the Plotkin bounds states that $N \leq \frac{d}{d-(1-q^{-1})L}$.

As the following theorem is the most well-known fact in strong traceability, a short proof is given.

Theorem 1.3.4 (Construction using ECC Chor et al. (2000)) *Suppose that $\mathcal{C}$ is an (L, N, q)-ECC having a minimum distance $d > L(1 - c^{-2})$. Then, $\mathcal{C}$ is a c-TA code.*

Proof. Assume collusion $\mathcal{C}_o$ has c colluders, and $\hat{\mathbf{m}} \in \text{desc}(\mathcal{C}_o)$.

First, we give a lower bound to $|\mathcal{I}(\hat{\mathbf{m}}, \mathbf{n})|$, $\mathbf{n} \in \mathcal{C}_o$. Assuming $A1$, the symbols of $\hat{\mathbf{m}}$ have been drawn from the symbols of the colluders. If the colluders share the risk evenly and if c divides L, then L/c symbols of $\hat{\mathbf{m}}$ come from the codeword of each colluder. As colluders may have the same symbol at index j, $|\mathcal{I}(\hat{\mathbf{m}}, \mathbf{n})|$ is at least equal to L/c. If c doesn't divide L, then we must take $\lfloor L/c \rfloor$. If the colluders doesn't share the risk evenly, some contribute less, others contribute more, so that, finally, there is at least a codeword $\mathbf{n}$ in $\mathcal{C}_o$ such that $|\mathcal{I}(\hat{\mathbf{m}}, \mathbf{n})| \geq \lfloor L/c \rfloor$.

Second, we give a lower bound to $|\mathcal{I}(\hat{\mathbf{m}}, \mathbf{m}_i)|$, $\mathbf{m}_i \in \mathcal{C} \backslash \mathcal{C}_o$. The distance between any two codewords (especially $\mathbf{n} \in \mathcal{C}_o$ and $\mathbf{m}_i \in \mathcal{C} \backslash \mathcal{C}_o$) is at least equal to d. Thus,

$|\mathcal{I}(\mathbf{n}, \mathbf{m}_i)| \leq L - d = Lc^{-2}$. We are sure that the symbols in common between $\mathbf{m}_i$ and $\hat{\mathbf{m}}$ are coming from the colluders' codewords. Hence, $\forall \mathbf{m}_i \in \mathcal{C} \backslash \mathcal{C}_o$,

$$\mathcal{I}(\hat{\mathbf{m}}, \mathbf{m}_i) \subseteq \bigcup_{\mathbf{n} \in \mathcal{C}_o} \mathcal{I}(\mathbf{n}, \mathbf{m}_i)$$

This gives $|\mathcal{I}(\hat{\mathbf{m}}, \mathbf{m}_i)| \leq \sum_{\mathbf{n} \in \mathcal{C}_o} |\mathcal{I}(\mathbf{n}, \mathbf{m}_i)| < c.Lc^{-2} = Lc^{-1}$.

Finally, $\mathcal{C}$ is a c-TA code because $\forall \hat{\mathbf{m}} \in \text{desc}(\mathcal{C}_o)$, $\forall \mathbf{m}_i \in \mathcal{C} \backslash \mathcal{C}_o$, $\exists \mathbf{n} \in \mathcal{C}_o$: $|\mathcal{I}(\hat{\mathbf{m}}, \mathbf{n})| \geq Lc^{-1} > \mathcal{I}(\hat{\mathbf{m}}, \mathbf{m}_i)$.

It is possible to extend theorem 1.3.4 with the case of e erasures:

Theorem 1.3.5 (Construction using ECC with erasures Safavi-Naini and Wang (2001)) *Under A4, suppose that $\mathcal{C}$ is an (L, N, q)-code having minimum distance $d > L(1 - c^{-2}) + ec^{-2}$. Then, $\mathcal{C}$ is a c-TA code.*

References Stinson and Wei (1998) and Hollmann et al. (1998) have independently suggested the use of Reed-Solomon codes. This is also the key idea in Fernandez and Soriano (2004b). For instance, suppose L, q and c are given, with q a prime power and $L \leq q + 1$. Then, there exists an (L, N, q) c-TA based on a RS code (length L, dimension $k = \lceil L/c^2 \rceil$, minimum distance $d = L - k + 1$) in which $N = q^{\lceil L/c^2 \rceil}$.

q-ary alphabet Boneh and Shaw show that strong traceability is not possible for $c \geq 2$ with binary alphabet. This implies the use of q-ary alphabet with $q > 2$ when we expect more than two colluders. Staddon et al. generalized this statement in the following theorem:

Theorem 1.3.6 (Collusion's size, Alphabet's size Staddon et al. (2001)) *If $\mathcal{C}$ is a (L, N, q) code with $N - 1 \geq c \geq q$, then $\mathcal{C}$ is not c-IPP.*

For c-TA codes based on ECC (according to theorem 1.3.4), we have the following theorems due to the Plotkin bound.

Theorem 1.3.7 (Minimal size of alphabet $\mathcal{Q}$ for c-TA codes Safavi-Naini and Wang (2001)) *Let $\mathcal{C}$ be an (L, N, q)-ECC with d satisfying theorem 1.3.4. If $N > q > 2$, then $q > c^2$.*

Theorem 1.3.8 (Extension to the case of erasures Safavi-Naini and Wang (2001)) *Under A4, let c and e be integers, and $\mathcal{C}$ be an (L, N, q)-ECC with d satisfying theorem 1.3.5. If $N > q > 2$, then $q > c^2 L/(L - e)$.*

Theorems 1.3.2 to 1.3.8 show that strong traceability is a hard constraint implying long codes and/or large alphabets. It is feasible in practice with Reed-Solomon codes (next subsection details a tracing algorithm at low complexity) but at the cost of a large alphabet: $q = \max(c^2 + 1, L - 1)$. Moreover, achievable rates defined by $\log_q N/L$ are in $1/q$; hence, they tend to zero as the number of users increases. Recent works use code concatenation (see subsection 1.3.7) to cope with large number of users and they managed to reach rates bounded away from zero Barg and Kabatiansky (2004).

List decoding The problem with fingerprinting decoding is the identification of several colluders. The input is a corrupted codeword, the output a list of codewords. This functionality is quite unusual in coding theory.

For c-TA short codes, the exhaustive decoding is practicable: we have to measure the N distances $d(\hat{\mathbf{m}}, \mathbf{m}_i)$ and to isolate the smallest ones. For c-IPP codes, the exhaustive decoding is quickly not viable: we have to build a lookup table associating the $\binom{N}{c}$ collusions with their set of descendants. The decoding then consists in finding in this table the sets of descendants containing $\hat{\mathbf{m}}$ and to calculate the intersection of their associated collusions. The size of the lookup table is quickly prohibitive when N increases.

The rest of the section presents an efficient tracing algorithm for c-TA built on ECC codes ($q > c^2$). The construction of strong traceability codes for $c+1 \leq q \leq c^2$, with an efficient decoding algorithm is an open-issue.

Usually, for a ECC code, if the number of errors e is greater than $\lfloor (d-1)/2 \rfloor$, there might be several codewords within distance e, and classical ECC fail to decode. Recent works dealing with list decoding from Guruswami and Sudan Guruswami and Sudan (1999) enforce this functionality. It succeeds in finding codewords within distance $L - \sqrt{(L-d)L}$ in polynomial time with L.

Theorem 1.3.9 (Reed-Solomon list decoding Silverberg et al. (2003)) *Let $\mathcal{C}$ be a Reed-Solomon code with $d > L(1 - c^{-2})$. There exist an efficient tracing algorithm for this c-TA code based on list decoding of and allowing to identify at least one traitor.*

List decoding also brings two other advantages to c-TA codes based on ECC Fernandez and Soriano (2004b); Silverberg et al. (2003):

- Decoding with side information about the channel transmission. Classical Reed-Solomon decoding algorithm do not take into account a priori information about the channel transmission. When a received symbol is incorrect, there is no advantage given to the other symbols. On contrary, list decoding algorithms use reliability measures about the received symbols. In a fingerprinting problem, it might be possible to give the reliability that the mark is undetected (the received symbol is correct) or, on contrary, to give reliability to colluders' symbols in that detected mark. List decoding takes into account side information (this has a different meaning than in chapter) about the collusion attack model and the embedding technique.

- Computationally efficient decoding. List decoding complexity of Reed-Solomon, for instance, are in polynomial time of L, or equivalently, a polynomial time in $c \log N$. This is slightly better than the exhaustive search in $O(N)$.

Iterative decoding List decoding has the following drawback. It works great when colluders share evenly the risk, ie. when $|\mathcal{I}(\hat{\mathbf{m}}, \mathbf{m}_i)| = cst, \forall \mathbf{m}_i \in \mathcal{C}_o$. But, if one colluder contribute less than the others, list decoding doesn't succeed to find him. This is not a critical point as at least one colluder is caught. Fernandez and Soriano propose an elegant way to deal with the situation where colluders' contributions are uneven Fernandez and Soriano (2004b). First, they prove the following theorem:

Theorem 1.3.10 (Iterative decoding Fernandez and Soriano (2004a)) *Under A1, Let $\mathcal{C}$ be a c-TA (L, N, q)-code with minimum distance d. Suppose that $j < c$ already identified positive parents jointly match less than $L - (c - j)(L - d)$ symbols of $\hat{\mathbf{m}}$. Then, any codeword that matches with $(c - j)(L - d) + 1$ of the unmatched positions is also a positive parent.*

Under A4, suppose that $j < c$ already identified positive parents jointly match less than $L - e - (c - j)(L - d)$ symbols of $\hat{\mathbf{m}}$. Then, any codeword that matches with $(c - j)(N - 1) + 1$ of the unmatched positions is also a positive parent.

This shows that when the dominating contributors of the collusion are caught, it is still possible to find the remaining colluders with a list decoding. In their algorithm, a first list decoding succeeds to find $j < c$ colluders. The received word is modified, creating a copy $\hat{\mathbf{m}}^{(1)}$ of $\hat{\mathbf{m}}$ where symbols matching with caught colluders codewords are erased. A list decoding is performed on this new vector. This process is iterated until c colluders are caught or $\hat{\mathbf{m}}^{(k+1)} = (*, \ldots, *)$.

$$\hat{m}^{(k+1)}[j] = \begin{cases} * & \text{if } j \in \bigcup_{\text{caught } \mathbf{m}_i \in \mathcal{C}_o} \mathcal{I}(\hat{\mathbf{m}}, \mathbf{m}_i) \\ \hat{m}[j] & \text{else.} \end{cases}$$

1.3.7 Weak traceability

Strong traceability implies two big drawbacks: long codes over large alphabets. This stems in a lack of feasibility for some applications. This is the reason why weak traceability is sometimes preferred. The tracing algorithm usually allows the capture of only one colluder with an error probability ϵ exponentially decreasing as the code length increases. The code is said to be c-secure with ϵ-error. This new parameter renders the study of weak traceable codes more complex.

This brings a new concept in fingerprint that we have avoided so far. This probability of error ϵ introduces the notion of randomness. What is random here? The marking assumption states that colluders are able to locate the detectable marks. This doesn't imply any knowledge about the scheme. But, the colluders may or may not be able to read the embedded symbol or to write in place a given symbol. They might simply be able to change the symbol to another one which they do not control. In the same way, they may or may not know their codewords or the codewords not in the collusion.

These issues are related to their knowledge about the fingerprinting scheme. To prevent this, the scheme usually uses some secret keys. If the technique which embeds the symbols in the content uses a secret key [5], this prevents the pirates to have a read/write access to the hidden channel. In the same way, the code might be dependent on a secret key. This is the way we maintain traitors in a blind state: they do not know what they are actually forging as $\hat{\mathbf{m}}$. This is the reason why we assume that $\hat{\mathbf{m}}$ is a random word uniformly distributed in $\text{desc}_c(\mathcal{C})$.

In this chapter, we focus on binary weak traceable fingerprinting codes because they are the most experienced in literature, and we illustrate typical construction

[5] We assume here that this key remains secret even if several contents are fingerprinted with it.

through some examples. For a more developed study, see the technical report of H. G. Schaathun who has compared performances of many schemes Schaathun (2004).

Tools for codes construction

Permutation Let us start with the following $(3, 4, 2)$-code, denoted $\mathcal{C}(4, 1)$:

$$\begin{aligned}\mathbf{m}_1 &= (1,1,1)\\ \mathbf{m}_2 &= (0,1,1)\\ \mathbf{m}_3 &= (0,0,1)\\ \mathbf{m}_4 &= (0,0,0).\end{aligned}$$

This code is not even 2-frameproof as users 1 and 3 can easily frame user 2. However, if these dishonest users know nothing about the embedding technique and the code, they pick at random a word in $\text{desc}(\{\mathbf{m}_1, \mathbf{m}_3\}) = \{(1,1,1); (0,0,1); (1,0,1); (0,1,1)\}$. As pirates are not stupid, they will not choose the first two words. Even if their knowledge is reduced to the minimum, we can argue that they certainly notice that these words correspond to their codewords. Hence, the probability to frame user 2 with the last word is 1/2. If they choose $\hat{\mathbf{m}} = (1,0,1)$, the tracing system accuses user 1 without any doubt (because the first symbol is set to '1').

Repeating this analysis for each collusion of size 2, we come with the following expectations: the probability to frame an innocent is equal to 2/9, else, the tracing algorithm finds a traitor with probability equal to 1. Actually, this code is not completely bad. The probability of framing an innocent slowly vanishes when N increases. However, for a given number of users, it would be great to lower this probability to a fixed level. A repetition code is used for this purpose in Boneh and J. Shaw (1998). Consider the code $\mathcal{C}(N, R)$ which is a $(R(N-1), N, 2)$-code, build by replicating R times each column of code $\mathcal{C}(N, 1)$. For instance, here is the code $\mathcal{C}(4, 3)$:

$$\begin{aligned}\mathbf{m}_1 &= (1,1,1,1,1,1,1,1,1)\\ \mathbf{m}_2 &= (0,0,0,1,1,1,1,1,1)\\ \mathbf{m}_3 &= (0,0,0,0,0,0,1,1,1)\\ \mathbf{m}_4 &= (0,0,0,0,0,0,0,0,0).\end{aligned}$$

Once again, the main issue depends whether pirates can notice the repetition. If this is the case, they will change R consecutive detectable marks in the same way and the repetition does not help. A permutation $\pi(.)$ defined on $\{1, \ldots, R(N-1)\}$ serves as a secret key. This permutation scrambles the symbol positions before embedding them in content. Hence, dishonest users cannot notice repeated symbols (unless they notice only R detectable marks). The inverse permutation is applied just after symbol decoding to get back to code $\mathcal{C}(N, R)$.

Notice that if a symbol '1' (resp. '0') is found in the R first (last) positions, then user 1 (user N) is guilty, without any doubt. See also that if user s, $1 < s < N$, is innocent, the collusion cannot distinguish columns $\{R(s-2)+1, \ldots, R(s-1)\}$ from $\{R(s-1)+1, \ldots, Rs\}$. The only strategy of the collusion is to randomly pick the symbols in these locations. Boneh and Shaw founded a tracing algorithm on these observations, which allows the following theorem. In addition, their code tackles

erasures (assumption $A3$), by replacing symbol '$*$' by symbol '0' before the word is processed by the tracing algorithm.

Theorem 1.3.11 (Replication with permutation Boneh and J. Shaw (1998)) *For $N \geq 3$ and $1 > \epsilon > 0$, let $R = 2N^2 \log(2N/\epsilon)$. Code $\mathcal{C}(N, R)$ has the following property: Whatever the size of the collusion, the tracing algorithm accuses at least one user and the probability of accusing an innocent is lower than ϵ. We say this code is N-secure with ϵ-error. Its length is $L = O(N^3 \log(N/\epsilon))$.*

Concatenation The code above is not practical as its length grows roughly with the number of users to the power of three. The most well known tool used in weak traceability in order to tackle big number of users is the concatenation of codes. Let $\mathcal{C}_I$ be a $(L_I, q, 2)$-code, called the inner code. Let $\mathcal{C}_O$ be a (L_O, N, q)-code, called the outer code. The code $\mathcal{C}$ resulting from the concatenation $\mathcal{C}_I \circ \mathcal{C}_O$ is a $(L_I L_O, N, 2)$-code. The construction of $\mathcal{C}$ is quite simple. We consider the inner code as a one-to-one mapping from $\mathcal{Q}_O$ (isomorphic to $\{1, 2, \ldots, q\}$) to the subset $\mathcal{C}_I \subset \mathcal{Q}_I^{L_I}$ (here, $\mathcal{Q}_I = \{0, 1\}$). Hence, in the codewords of $\mathcal{C}_O$, one can replace symbols of $\mathcal{Q}_O$ by codewords of $\mathcal{C}_I$. These codewords are called blocks Schaathun (2004), coordinates Barg et al. (2003), or components Boneh and J. Shaw (1998). Finally, this forms N binary sequences of length $L_I L_O$.

The decoding of such a concatenated code begins by decomposing the received word in blocks of L_I bits. Then, the decoding for the inner code is applied. This gives, for each block, a codeword or a set of codewords of $\mathcal{C}_I$. This sequence of L_O results is the input for the decoding for the outer code.

At the encoding side, the outer code is called first, then the inner code. In the tracing algorithm, the inner code is decoded first, then the outer code; whence the terminology 'outer/inner'.

The basic idea is that the inner code tackles the robustness against collusion of size c, but for a low number of users. The concatenation with the outer code allows to increase the number of users while keeping the property of the inner code. Yet, this kind of scheme is quite hard to fine tune. If we use a good inner code, with a low probability of error (or even null if its a strong traceable code), the blocks are already quite long. There is clearly a tradeoff to strike between the features of the inner and outer codes. We will not study this problem in general, but we examine two examples.

Example 13 (Outer random code Boneh and J. Shaw (1998)) *Let $\mathcal{C}_O$ be an (L_O, N, q)-code where the codewords are chosen independently and uniformly random. Let $\mathcal{C}_I$ be the code $\mathcal{C}(N_I, R)$ of theorem 1.3.11. Set $q = N_I = 2c$, $\epsilon_I = 1/2c$, so that $R = 8c^2 \log(8c^2)$. And, for the outer code, $L_O = O(c \log N/\epsilon)$. The concatenated code is then c-secure with ϵ-error, with a length $L = O(c^4 \log(N/\epsilon) \log c)$ Schaathun (2004).*

We do not prove the performances of this code, we only give insights to calculate the error probability. The tracing algorithm decodes the inner code first: for each block i, it gives a codeword $\mathbf{m}[i] \in \mathcal{C}_I$. This codeword matches one of the codeword of the colluders with an error probability ϵ_I. The outer code is decoded looking in $\mathcal{C}_O$ for the nearest codeword to $\hat{\mathbf{m}} = (\mathbf{m}[1], \ldots, \mathbf{m}[L_O])$.

As codewords of $\mathcal{C}_O$ are statistically independent, the codewords not in the collusion are independent from $\hat{\mathbf{m}}$. The probability that one of these matches $\hat{\mathbf{m}}$ for a given block is $1/q$. This means that the number of blocks $r(\text{innocent})$ where the codeword of an innocent matches $\hat{\mathbf{m}}$ is a binomial variable of L_O trials with probability $1/q$. Its expectation is then $L_O/q = L_O/2c$.

On the other hand, the inner decoding succeeds in a number of block which is a binomial variable with L_O trials and probability $1-\epsilon_I$. One codeword of the collusion matches $\hat{\mathbf{m}}$ in at least one out of c blocks. This corresponds to the worse case where colluders share evenly the risk. This means that the number of blocks $r(\text{traitor})$ where the nearest codeword of traitors matches $\hat{\mathbf{m}}$ is a binomial variable of $L_O(1-\epsilon_I)$ trials with probability $1/c$. Its expectation is then $L_O(1-\epsilon_I)/c$.

Finally, the analysis results in choosing L_O sufficiently big such that the event $r(\text{innocent}) > r(\text{traitor})$ happens with a probability lower than ϵ.

Example 14 (Outer Reed-Solomon code Barg et al. (2003)) *Let $\mathcal{C}_O$ be a (L_O, N, q)-code as defined in theorem 1.3.5, with a minimum distance such that*

$$d > L_0 \left(1 - \frac{1}{c^2} + \frac{c-1}{c(q-1)}\right). \tag{1.12}$$

The authors of Barg et al. (2003) have chosen a Reed-Solomon code. Let $\mathcal{C}_I$ be a c-SFP $(L_I, q, 2)$-code. The concatenated code is then a c-secure with an error probability bounded by (Schaathun (2004)):

$$\epsilon < N(q-2)^{-L_O(1-6(1-\delta))}. \tag{1.13}$$

The decoding is not trivial. First, there is no efficient decoding (and even construction) of c-SFP codes with $c \geq 3$. Usually, the only way is to build a look-up table: for the $\binom{q}{c}$ c-sets $\mathcal{C}_{Io}$, associate $\text{desc}(\mathcal{C}_{Io})$. The decoding of the inner code then parses the lookup table until it finds the block $\hat{\mathbf{m}}[i]$ in a set $\text{desc}(\mathcal{C}_{Io}[i])$. The output is not a single block (ie., a codeword of $\mathcal{C}_I$) but a collusion $\mathcal{C}_{Io}[i]$. This technique is only manageable for small c. In the second stage, the goal is to find the nearest codeword in $\mathcal{C}_O$ from the set $\mathcal{H} = \{\hat{\mathbf{m}} \in \mathcal{Q}_O^{L_O} | \hat{\mathbf{m}}[i] \in \mathcal{C}_{Io}[i]\}$. Outer codes like Reed Solomon allows an efficient solution for this task with the list decoding algorithm (see subsection 1.3.6).

Performances

As mentioned above, the codes from example 14 can only tackle small collusion size. Table 1.1 gives the features of some practical examples. Notice that the length of the code roughly follows $\log N$ whereas it exploses with c. Codes based on example 13 are longer as showed in table 1.2. However, it is easier to cope with bigger collusions at a low complexity cost.

1.3.8 Fingerprinting multimedia content

When dealing with multimedia content, the main difficulty is to imagine all the actions the colluders can do. Each of them receives different pieces of content watermarked with the same fingerprint. There is thus a threat that a dishonest user alone can

[htdp]

N users	length L	c traitors	error prob. ϵ
2^{28}	1 386	2	$0,23.10^{-12}$
2^{42}	4 919	2	$0.14.10^{-10}$
2^{14}	111 872	3	$0,32.10^{-33}$
2^{14}	49 818	3	$0,61.10^{-10}$
2^{21}	66 424	3	$0,63.10^{-10}$
2^{32}	80 606	3	$0,55.10^{-10}$

Table 1.1: A family of codes built upon example 14. Results are taken from Schaathun (2004).

[htdp]

N users	length L	c traitors	error prob. ϵ
2^{10}	299 889	2	1.10^{-10}
2^{20}	367 359	2	1.10^{-10}
2^{30}	435 471	2	1.10^{-10}
2^{10}	$4,78.10^{8}$	10	1.10^{-10}
2^{20}	$6,56.10^{9}$	20	1.10^{-10}
2^{30}	$4,16.10^{10}$	30	1.10^{-10}

Table 1.2: A family of codes built upon example 13. Results are taken from Schaathun (2004).

gain information about the watermarking technique, the secret key and/or his user fingerprint. All together, they can have several pieces of the same content but watermarked with their different fingerprint. Once again, information about the technique or its secret parameter certainly leaks from these data. Yet, one usually considers the colluders as not so smart pirates. The only one attack, envisaged so far, is very simple: colluders mix these different copies of the same content into one pirated version. They hope that this will jam the fingerprints into a non-meaningful decoded message which might delude the system.

Multimedia documents are more usually considered as signals (see chapter 2) than binary sequences and the model of Boneh and Shaw (ie., the marking assumption, see 1.3.4) is not appropriate. From now on, content are considered as real signals and fingerprinting consists in the addition of a watermark signal. The original content is represented by $\mathbf{x} \in \mathbb{R}^{L_X}$. User i receives his copy $\mathbf{y}_i = \mathbf{x} + g\mathbf{w}_i$, with g a scalar fixing the embedding strength and $\mathbf{w}_i$ the fingerprint signal such that $\|\mathbf{w}_i\|^2 = L_X$.

Independent watermark signals

In this section, watermark signals are orthogonal pseudo-random sequences statistically independent upon the original content. Their orthogonality implies that the figure of users N is less than L_X.

The colluders mix their different copies into one version $\mathbf{y}$. This process is often considered as an average attack[6]: $\mathbf{y} = \sum_{\ell=1}^{c} \mathbf{y}_{i_\ell}/c$, where $\{i_1, \ldots, i_c\}$ are the indices of c colluders. Indeed, this is the special case of a much more general model where the mixing is $\mathbf{y} = \sum_{\ell=1}^{c} \mathbf{l}_\ell \otimes \mathbf{y}_{i_\ell}$, with $\otimes$ being the samplewise multiplication and $\{\mathbf{l}_\ell\}$ being c weighing vectors such that $\sum_{\ell=1}^{c} \mathbf{l}_\ell = \mathbf{1}_{L_X}$. This model describes a lot of attacks, including (the definitions being given for all $(\ell, j) \in \{1, ..., c\} \times \{1, ..., L_X\}$):

- Average: $l_\ell[j] = 1/c$.
- Mosaic: $l_\ell[j] = \Pi_{a_\ell, b_\ell}[j]$ where $\Pi_{a,b}$ is the indicating function of interval (a, b), with $a_\ell = (\ell - 1)L_X/c + 1$ and $b_\ell = \ell L_X/c$ (we assume that $\exists k \in \mathbb{N} | L_X = kc$ for simplicity).
- Minimum: $l_\ell[j] = 1$ if $\ell = \arg\min_i y_{i_\ell}[j]$, else 0.
- Maximum: $l_\ell[j] = 1$ if $\ell = \arg\max_i y_{i_\ell}[j]$, else 0.
- MinMax: $l_\ell[j] = 1/2$ if $\ell \in \{\arg\min_\ell y_{i_\ell}[j], \arg\max_\ell y_{i_\ell}[j]\}$, else 0.
- Randomized: $l_\ell[j] = 1$ if $\ell = \arg\max_\ell y_{i_\ell}[j]$ and $r[j] = 1$, or if $\ell = \arg\min_\ell y_{i_\ell}[j]$ and $r[j] = 0$, else 0 ($r[j]$ being a binary random variable).

Colluders may also distort this mixed signal with some noise $\mathbf{z}$ in order to delude the tracing.

The protection system observes signal $\mathbf{v} = \mathbf{y} + \mathbf{z}$. N tested statistics are calculated as correlations of this received signal with the N orthogonal watermark signals: $t_k = \langle \mathbf{w}_k, \mathbf{v} \rangle$ $1 \leq k \leq N$. Zhao et al. (2003) claim that the Z-statistic (which is a

[6] whence the confusion between the terms collusion and average attack.

deterministic function of the normalized correlation) is more robust to attack, but we deal with this less complex statistic to keep the study simple in this chapter.

The impact of the collusion has the following expression:

$$t_k = \langle \mathbf{w}_k, \mathbf{x} \rangle + \langle \mathbf{w}_k, \mathbf{z} \rangle + g \sum_{\ell=1}^{c} \langle \mathbf{w}_k, \mathbf{l}_\ell \otimes \mathbf{w}_{i_\ell} \rangle = n_k + d_k. \tag{1.14}$$

The first two terms are considered as independent Gaussian centered variables with variances equals to $L_X \sigma_X^2$ and $L_X \sigma_Z^2$. We gather them into one random variable denoted by N_k.

There is a big disagreement in the fingerprinting community concerning this noise. Some assume that the detection process is not blind: the protection system knows the original piece of content. The first term shall then be removed. The tested statistic has then a far smaller variance and the detection process is thus more accurate. Others think that it is too complex for the protection system to find back the original content in a database, to perfectly synchronize the pirated version with this original (for instance, movies must be temporally and spatially synchronized) and to finally cancel the contribution of the original. It is true that a part of the problem is solved if these tasks are done manually or semi-automatically. This issue depends on the application, and people work with blind detectors to free their study from hazardous assumptions on this subject. Here, we gather the first terms into one random variable $N_k \sim \mathcal{N}(0, \sigma_N^2)$ where $\sigma_N^2 = L_X(\sigma_Z^2 + \delta \sigma_X^2)$ with $\delta = 1$ in the blind framework, $\delta = 0$ in the non blind framework.

Random variable D_k tackles the last term of equation (1.14). Invoking the independence between the weighting functions and the orthogonal watermark signals, its expectation simplifies to:

$$\mu_{D_k} = E[D_k] = g \sum_{\ell=1}^{c} \sum_{j=1}^{L_X} l_\ell[j] E[w_k[j] w_{i_\ell}[j]] = g \sum_{\ell=1}^{c} \delta_{k,i_\ell} \sum_{j=1}^{L_X} l_\ell[j]. \tag{1.15}$$

Thus, D_k is centered if user k is not a colluder. It means, in a way, that this scheme is frameproof as no strategy allows the colluders to frame a honest user. In the alternative hypothesis, user i_ℓ is a colluder and the term $\sum_{j=1}^{L_X} l_\ell[j]$ is interpreted as the amount of the remaining part of watermark signal $\mathbf{w}_{i_\ell}$. Note that:

$$\sum_{\ell=1}^{c} \sum_{j=1}^{L_X} l_\ell[j] = \sum_{j=1}^{L_X} \sum_{\ell=1}^{c} l_\ell[j] = L_X. \tag{1.16}$$

As the colluders certainly share the risk evenly, one can assume that $|\sum_{j=1}^{L_X} l_\ell[j]| = cst, \forall \ell \in \{1, \ldots, c\}$. As there is no point for the colluders to have a negative correlation (the protection system would just consider absolute value of correlations), cst equals L_X/c according to the last equation. Consequently, $\mu_{D_{i_\ell}} = gL_X/c$ whatever the attack. In a way, this justifies the overwhelming consideration for the average attack in the research works because more complex mixing models actually result in the same expectation of D_{i_ℓ}. However, the variance of D_{i_ℓ} depends on the mixing model and

it is much more difficult to analyze. Zhao et al. (2003) have presented experimental works about this subject. It appears that the different attacks lead to almost the same value of $\sigma^2_{D_{i_\ell}}$, except a very noticeable increase for the 'Randomized' strategy with $P_R(r[j]=1) = P_R(r[j]=0) = 1/2$. This is not surprising as the mixed signal becomes the original signal plus a very disturbed signal. For bounded fingerprint signals such that $|w_\ell[j]| < B$, this tends to the bounded signal with the highest power as the figure of colluders increases: $y[j] = x[j] \pm gB$. Note that this strategy also leads to the most distorted pirated content.

The collusion increases the number of correlation peaks by a factor c, but their values is decreased by a factor $1/c$. The collusion succeeds if the correlation peaks is below a decision threshold η. This threshold is calculated considering the probability of false alarm P_{fa}, which is the probability of accusing the honest user k: $P_{fa} = Pr(t_k > \eta) = Q(\eta/\sigma_N)$ (the definition of the Q-function and the calculus are given in section 2.2.2). The probability of identifying colluder i_ℓ is indeed the power of the detection test: $P_p = Pr(t_{D_{i_\ell}} > \eta) = Q((\eta - \mu_{D_{i_\ell}})/\sqrt{\sigma^2_N + \sigma^2_{D_{i_\ell}}})$. As $Q(.)$ is a decreasing function, colluders wish to decrease $\mu_{D_{i_\ell}}$ increasing the figure of colluders or to increase $\sigma_{D_{i_\ell}}$ with the 'Randomized' mixing.

The global probability to identify at least one colluder equals to $1-(1-P_p)^c$. The scheme has not the strong traceability property due to the probability of false alarm. Note that the probability given above is for one user. The total probability of false alarm is equal to $1-(1-P_{fa})^{N-c} \sim (N-c)P_{fa}$. Consequently, the system is c-secure with ϵ-error, $\epsilon < (1-P_p)^c + (N-c)P_{fa}$.

This fingerprinting scheme has two drawbacks: a relatively high total probability of false alarm and a complexity proportional to the number of users as N correlations are calculated. Trappe et al. (2003) propose a recursive algorithm to reduce the number of correlations. Suppose we have only one colluder. $n-1$ correlations will give small results centered on zero, one correlation will give a large positive value. Their algorithm take benefit of the linearity of the correlation process: $\langle \sum_k \mathbf{w}_k, \mathbf{x} \rangle = \sum_k \langle \mathbf{w}_k, \mathbf{x} \rangle$. Let us group in a vector $\mathbf{s}_1$ the sum of the first $N/2$ fingerprint signals, and in a vector $\mathbf{s}_2$ the sum of the remaining fingerprint signals. We calculate the correlations $\langle \mathbf{s}_1, \mathbf{x} \rangle$ and $\langle \mathbf{s}_2, \mathbf{x} \rangle$. The one that is larger than the other indicates the group which contains the fingeprint $\mathbf{w}_{i_1}$. We split this group into two sets, we calculate the two new sum vectors $\mathbf{s}_1$ and $\mathbf{s}_2$ and their correlation with $\mathbf{x}$, and iterate the process until we disclose the index i_1. Figure 1.5 illustrates this process for 8 users. The number of correlations is at most $2(\lceil \log_2 N \rceil - 1)$. However, when there are several colluders, the number of correlations increases and the difference with the exhaustive search is less interesting. Moreover, it is difficult to assess the probability of false alarm and the power of the test of this iterative method. A full study is available in article Trappe et al. (2003).

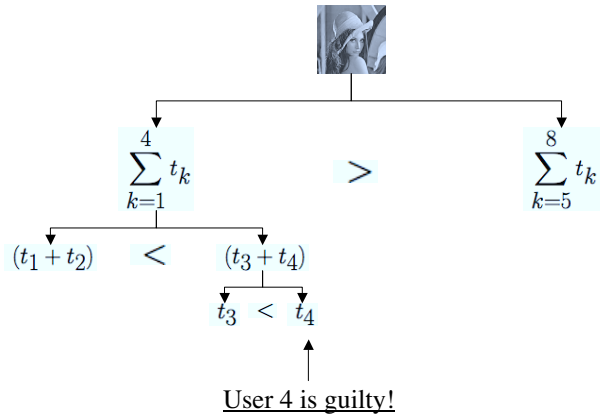

Figure 1.5: Iterative decoding of orthogonal fingerprint signals, with one dishonest user among 8 users. This algorithm only needs 6 correlations and 3 comparisons where the exhaustive search requires 8 correlations and 7 comparisons.

Redundant watermark signals

The fingerprint signals used in the last subsection can be formulated as the modulation of orthogonal careers by binary messages $\{\mathbf{b}_k\}$:

$$\mathbf{w}_i = \sum_{k=1}^{L_b} m(b_i[k])\mathbf{u}_k, \tag{1.17}$$

provided that the modulation is a On-Off Keying (i.e., $m(b) = 1$ if $b = 0$, 0 else) and the messages are codewords of the N-FP $(N, N, 2)$-code $\Gamma^{(6)}$. This technique follows the strategy of embedding cryptographic codes in multimedia content thanks to a watermarking technique. Having established this link, one can observe that the On-Off Keying is not the best modulation. A BPSK modulation is more powerful (i.e., $m(b) = 1$ if $b = 1$, -1 else). This new scheme manages $N < 2^{L_b}$ users with only L_b careers. It is efficient in the sense that it addresses more users than the number of careers. Indeed, shorter codes allow greater power per career g (for a fixed embedding distortion) and also need less correlation calculus at the decoding side. The ratio N/L_b is sometimes used as an efficiency criterion Trappe et al. (2003).

Having selected a modulation (or more generally a watermarking technique), the next step is to model the impact of the collusion attack on the message decoding. The main difference compared with the last subsection is that we have a watermark decoder (that decodes bits) instead of a watermark detector (that looks for the presence (On) or the absence (Off) of watermark signals). For instance, if users 1 and 2 collude to make the average attack $\mathbf{y} = \mathbf{x} + g(\mathbf{w}_1 + \mathbf{w}_2)/2$, careers transmitting different bits will vanish because $m(0) + m(1) = 0$, whereas the others will sustain their information bits. The situation is trickier with several colluders. In general, the

pirated content is expressed by:

$$\begin{aligned}\mathbf{v} &= \mathbf{x}+\mathbf{z}+\frac{g}{c}\sum_{k=1}^{L_b}\sum_{\ell=1}^{c} m(b_{i_\ell}[k])\mathbf{u}_k \\ &= \mathbf{x}+\mathbf{z}+g\sum_{k=1}^{L_b} m_k\mathbf{u}_k, \end{aligned} \tag{1.18}$$

with $m_k \in \{-1, -(c-1)/c, \ldots, (c-1)/c, 1\}$.

For a figure of colluders in the order of ten, it is illusive to base a traitor tracing program on the accurate estimation of the state of m_k due to the interference with the host and noise signals. This is the reason why Trappe et al. (2003) only detect the symbol $m_k = 1$ thresholding the correlations t_k with a quite high treshold $\eta = 0.9E[t_k|m_k = 1]$. Correlations smaller than this threshold are decoded as '0'. This looks like the way Boneh and Shaw replace erasures by symbols '0' (see subsection 1.3.7). The decoded bit $\hat{b}[k]$ is then a '1' if and only if all the embedded bits $\{b_{i_\ell}[k]\}$ were set to '1'. Formally, we have $\hat{b}[k] = \bigotimes_{\ell=1}^{c} b_{i_\ell}[k]$.

Trappe et al. assume that the impact of the collusion approximates the bitwise AND operation in the message domain Trappe et al. (2003). They then look for a AND-ACC (AND Anti Collusion Code) binary code. A trivial code is $\{110, 101, 011\}$. The AND operation on two codewords leaves two '0' indicating the colluders. Nevertheless, it is as inefficient as the independent watermark signals fingerprints scheme of subsection 1.3.8 because its ratio N/L_b equals 1. These authors create more powerful AND-ACC based on Balanced Incomplete Block Design, a mathematical tool which is out of scope in this tutorial chapter. They achieve a ratio N/L_b in $O(\sqrt{N})$. They also improve the decoding algorithm trying to maximize the extraction from the correlation of information about the colluders with a soft decoding. Experimental measurement of probability of false alarm and power of the test against the figure of colluders and the total number of user are given in their article Trappe et al. (2003).

Although that AND-ACC are based on an average attack model, we have seen in subsection 1.3.8 that more complicated attacks have the same impact because the colluders evenly share the risk. However, AND-ACC have never been tested against more malicious attacks using weighting vector such that those where $\sum_{j=1}^{L_X} l_\ell[j] < 0$ as presented in Wu (2005).

Example 15 (Malicious attack with linear combination) *Imagine a fingerprinting scheme using a BPSK watermarking technique with the following* $(4, 4)$*-code:* $\mathcal{C} = \{1110, 1101, 1011, 0111\}$*. Users 1, 2, and 3 are the colluders. In a first case, they create:* $\mathbf{y} = (\mathbf{y}_1+\mathbf{y}_2+\mathbf{y}_3)/3 = \mathbf{x}+\mathbf{u}_1+(\mathbf{u}_2+\mathbf{u}_3+\mathbf{u}_4)/3$*. The protection system decodes* $\mathbf{y}$ *in* $\hat{\mathbf{m}} = (1000)$ *and it concludes that the colluders are users 1, 2, and 3. In a second case, colluders create:* $\mathbf{y} = \mathbf{y}_1+\mathbf{y}_2-\mathbf{y}_3 = \mathbf{x}+\mathbf{u}_2+(\mathbf{u}_1-\mathbf{u}_3-\mathbf{u}_4)/3$*. This attack completely escapes from the the bitwise AND model. This signal is decoded in* (0100)*, and the protection system accuses users 1, 2 and 4, who is not guilty!*

1.3.9 Conclusion

This chapter has presented the main issues concerning fingerprinting. The designer usually starts with a model of the collusion attack: cryptographers use a version of the marking assumption whereas Trappe and al. go for the AND model. After the conception of a code which is c-frameproof for this model, the designer studies the probability of false alarm P_{fa} and the power of the test, $P_p = 1 - \epsilon$, depending on the figure of colluders.

We have seen that the difficulties of fingerprinting essentially reside in the complexity of the traitor tracing algorithm especially when the number of users is huge, and the conception of a model tackling all possible collusion attacks.

From a watermarking point of view, the most significant references are the works from W. Trappe and M. Wu:Trappe et al. (2003); Wu et al. (2004). From a cryptographic point of view, accessible references (for non expert) are Fernandez and Soriano (2004b) for strong traceability and Schaathun (2004) for weak traceability.

1.4 Watermarking protocols

This section considers protocols between two or more collaborating parties using the watermarking primitive. A protocol is an algorithm defining a sequence of actions required to the parties in order to accomplish a specified objective (see full definition in Menezes et al. (1996)). The main focus is watermarking embedding between a buyer and a seller, and watermarking decoding between a prover and a verifier.

1.4.1 Watermarking embedding protocol

There is only one kind of watermarking embedding protocol. The framework is a transaction between a buyer and a seller. Section 1.3 has already introduced the fact that the seller embeds a fingerprint in the image to trace dishonest buyers. However, in this section, the buyer also suspects the seller. For instance, a dishonest seller can give copies of the buyer's fingerprinted image to accomplices in charge of publishing it. Then, he feigns to discover these unauthorized uses. The pretendedly spoiled seller finally frames the innocent buyer.

Finally, one question sums up the problem: Where does the fingerprint embedding take place? It must not be located at the buyer's side because the seller does not want to give the original image. It must not be located at the seller's side because the buyer doesn't want to give the seller the access to the fingerprinted content.

Homomorphic encryption scheme

Before presenting the protocol, we detail the key idea which resorts to the homomorphism property of some cryptographic functions. We say that function $F(.)$ ranging from $\mathcal{X}$ to $\mathcal{Y}$ is homomorphic if it satisfies the following relationship:

$$F(x_1) * F(x_2) = F(x_1 \odot x_2) \quad \forall (x_1, x_2) \in \mathcal{X}^2, \tag{1.19}$$

where $\odot$ and $*$ are two binary operators respectively defined on $\mathcal{X}$ and $\mathcal{Y}$.

Suppose that x_1 is the original content $\mathbf{x}$, that x_2 represents the watermark signal $\mathbf{w}$, and that $\odot$ is the watermark embedding operator. Suppose also that function $F(.)$ is indeed an asymmetric k-keyed encryption $E(., k)$: one encrypts with public key k, and the decryption key is private and belongs to the buyer. With all these assumptions, we are ready to sketch a solution. The buyer and the seller encrypt the watermark signal and the original image respectively with the buyer's public key. The buyer sends the encrypted watermark $E(\mathbf{w}, k)$ to the seller. The seller performs the embedding in the 'encrypted domain': $E(\mathbf{y}, k) = E(\mathbf{x}, k) * E(\mathbf{w}, k)$. He sends the result to the buyer who decrypts with his private key to retrieve the fingerprinted picture. The seller has neither the fingerprint signal of the user nor the fingerprinted image. The buyer has never the original content.

Some technical details have to be discussed. First, we have to find a couple of asymmetric encryption scheme and watermarking technique which are compliant with respect to the binary operator $\odot$. For multimedia content, a quantization is required to have a discrete representation of the watermarking signal and the original image, which allows the application of cryptographic primitives only defined for integers. Moreover, these primitives usually work element by element.

Example 16 (Embedding from Memon and P.W. Wong (2001)) *This scheme uses the classical RSA crypto-system and a proportional embedding as proposed by Cox et al. (1997). The buyer selects two large primes p and q, and he computes $n = pq$, a private key d and the associated public key (e, n). He also generates a pseudo-random white gaussian noise which will be his secret signal $\mathbf{s}$. The signal $\mathbf{w} = \mathbf{1} + \alpha\mathbf{s}$ goes through a scalar quantizer, where α sets the embedding strength. Each sample is then encrypted with the public key and sent to the seller: $w_e[i] = \hat{w}[i]^e \ mod\ n$. The seller encrypts the quantized DCT coefficients of the original image: $x_e[i] = \hat{x}[i]^e \ mod\ n$. He interleaves the received encrypted watermark signal samples with a pseudo-random permutation $\pi(.)$, and computes $y_e[i] = x_e[i]w_e[\pi(i)] \ mod\ n$. Thanks to the homomorphic property ($\odot$ and $\star$ are both the multiplication in $\mathbb{Z}/n\mathbb{Z}$), the buyer receives samples $y_e[i]$ which are decrypted by $y[i] = y_e[i]^d \ mod\ n = x[i](1 + \alpha s[\pi(i)])$. It is up to him to de-quantize and to make the inverse DCT.*

However, homomorphic properties are not often desired for encryption schemes because an opponent observing two ciphertexts can produce a third one. Bit commitment scheme as exposed in example 17 are better examples of homomorphic cryptographic primitives. Second, the fact that the buyer knows the embedded signal is a security threat. There must be a way to include some secrecy in the process although the seller embeds the fingerprint signal in the 'encrypted domain'. The pseudo-random permutation $\pi(.)$ plays the role of the seller's secret key in example 16. With the marking assumption of subsection 1.3.4, the locations of the marks constitutes the seller's secret.

Example 17 (Embedding from Kuribayashi and Tanaka (2003)) *This scheme uses the Okamoto-Uchiyama bit commitment scheme with an additive embedding watermarking technique ($\odot$ is the addition in $\mathbb{Z}$ and $\star$ is the multiplication in $\mathbb{Z}/n\mathbb{Z}$). The buyer chooses two large primes p and q, computes $n = p^2q$, chooses $g \in \mathbb{Z}/n\mathbb{Z}$*

(see conditions in Menezes et al. (1996)), computes $h = g^n \bmod n$. *He creates his public key* (n, h, g) *and the related private key* (p, q). *The buyer also chooses a binary fingerprint signal* $\mathbf{w}$ *and a sequence* $\mathbf{a}$ *of random integers less than* n. *Each element is encrypted:* $w_e[i] = g^{w[i]}h^{a[i]} \bmod n$. *Alea* $a[i]$ *does not prevent the 'decryption' but it insures more security. The seller quantizes the* 8×8 *DCT coefficients with suitable quantization steps for all frequencies (like a quantization table of JPEG), and picks up a sequence* $\mathbf{b}$ *of random integers less than* n. *He 'encrypts' the quantized coefficient* $\hat{x}[i]$ *and sends it to the buyer:* $y_e[i] = g^{\hat{x}[i]}h^{b[i]} \bmod n$; *except that, at the secret marking positions, he sends* $y_e[i]w_e[i] \bmod n = g^{\hat{x}[i]+w[i]}h^{a[i]+b[i]} \bmod n$. *The buyer 'decrypts' with his private key and retrieves either* $\hat{x}[i]$ *either* $\hat{x}[i] + w[i]$. *It is up to him to de-quantize and to make the inverse DCT.*

The scheme of Pfitzmann and Schunter (1996) works in a similar way on binary original content with an homomorphic bit commitment based on quadratic residue and the Boneh and Shaw fingerprinting method. Locations of the marks constitute the seller's secret.

1.4.2 Buyer-seller protocols

The embedding of encrypted signals in encrypted content is the key idea of the buyer-seller protocols. However, it does not bring security on its own if not used in a protocol.

Watermark generation

The buyer is supposed to give the seller an encrypted fingerprint signal. However, as the seller cannot decrypt it, the buyer may send a fake signal. To prevent this hack, Pfitzmann and Schunter (1996) succeeds to invent a Zero-Knowledge protocol where the buyer proves to the seller that the encrypted fingerprint signal is valid. This works with the code of the Boneh and Shaw fingerprinting scheme for binary content of subsection 1.3.7. Another way is to resort to a trusted third party named watermark certification authority. This entity stores and encrypts with the buyer's public key the fingerprint signal in Memon and P.W. Wong (2001). In Kuribayashi and Tanaka (2003), it calculates cryptographic commitments which will convince the seller later on.

Traitor tracing

There are here two strategies. In the first case, the seller has a watermark decoder, which extracts the fingerprint. At least, the seller can re-encrypt this decoded message with his buyers' public key and he compares these ciphers to the encrypted messages he received from the buyers. Pfitzmann and Schunter (1996) succeeds to give a structure to the fingerprint such that the buyer's number appear in the clear. Kuribayashi and Tanaka (2003) are more oriented to anonymous fingerprinting, where the seller has to ask the registration center to disclose the identity of the buyer (for more details see Pfitzmann and Waidner (1997)).

In the second strategy, the fingerprinting scheme relies on a watermark detector as introduced in subsection 1.3.8. The seller cannot trace traitors as he does not know

the watermark signals of his clients. Memon and P.W. Wong (2001) propose that the seller embeds a first watermark on his own as presented in subsection 1.3.8, then he embeds the encrypted fingerprint provided by the buyer. The first one helps him tracing the buyer, the second one is used for the dispute resolution.

Dispute resolution

The dispute is a three party protocol between the seller, an arbiter and possibly the accused buyer. A deny of cooperation in order to prove his innocence, is considered as an evidence of guilt. In works like Pfitzmann and Schunter (1996) or Kuribayashi and Tanaka (2003), the dispute resolution is a Zero Knowledge protocol where the buyer has to prove that at least one fingerprint sample does not match with one element of the encrypted version he committed on at the beginning of the transaction. Memon and P.W. Wong (2001) have a much simpler protocol where the arbiter asks for the fingerprint signal in the clear and he compares its encrypted version with the message given to the seller at the beginning of the transaction. This has the drawback of disclosing the fingerprint, whence the arbiter must be a trusted third party which will not reveal this secret.

1.4.3 Difficulties

The issue of watermark security has not been tackled in these studies. The buyer receives a watermarked content, knowing the hidden message. This could certainly help the buyer (or a collusion of buyers) to disclose the seller's watermarking key. This is in a way equivalent to a Known Clear Text Attack in cryptanalysis.

The amount of data to be exchanged during the protocol can be very huge. First, the fingerprint codewords (or watermark signals) are usually very long. This is especially true if the watermark decoding is blind. Second, when encrypt them bit by bit (or respectively sample by sample), this multiplies this length by a factor of some thousands of bits (i.e., the size of one bit commitment). These protocols roughly needs some dozens of megabytes to be exchanged between the seller and the buyer. Recent works try to decrease this amount using encryption schemes with lower rate.

Finally, as certainly noticed by the reader, there are very few works concerning embedding protocols. They do not use recent advances in fingerprinting like the redundant watermarking signals presented in subsection 1.3.8. This idea is extremely challenging because the watermark signal is made from the seller's secret carriers and from the buyer's secret codeword.

1.4.4 Watermarking detection protocols

There are many issues about proving ownership with watermarking (Craver et al. (1998)). One of them takes place at the detection stage. A verifier has a piece of content and he wonders whether content is copyrighted. The prover has the detection key (usually he is the copyright holder). These two actors do not trust each other. If the prover gives the detection key, the verifier can then remove or add watermarks

without being authorized. If the verifier gives the suspicious picture, the prover might forge a fake detection key such that the detection test succeeds.

The basement of the watermark detection protocols is a well known cryptographic primitive: the zero-knowledge proof. The zero-knowledge watermarking detection aims at proving the presence of a watermark signal in a suspicious content without any leak of information about the detection key.

Cryptographic commitment

A very useful tool in zero-knowledge protocol is the cryptographic commitment. Everybody knows the game where a child has first selected an object in his mind, and a second child asks him questions to which he answers by only yes or no, in order to guess the object. The first child easily cheats changing his mind provided that the newly selected object matches his past answers. Indeed, at the beginning of the game the first child should *commit*: he writes the object's name on a piece of paper, folds it, and puts it in a visible place. Of course, the paper is folded so that the second child cannot see anything of the writing. At the end of the game, the child will unfold the paper proving that the first child has fairly played.

A cryptographic commitment scheme is composed of two protocols : $com(.)$ to commit to a value $m \in \mathcal{M}$ and $open(.)$ to open a commitment. This could be a kind of one-way function $f(.)$ from message space $\mathcal{M}$ to commitment space $\mathcal{C}$. $com(.)$ is simply the application of $f(.)$ to m by the committer: $c = f(m)$. In protocol $open(.)$, the committer reveals m and the verifier checks that $c = f(m)$. A commitment must have two properties:

- binding: a dishonest committer cannot forge a pre-image m' such that $m' \neq m$ and $com(m) = com(m')$.
- hiding: No information about m leaks from $com(m)$.

Commitment is needed in watermarking detection protocol to avoid that the prover forges a fake detection key. The commitment takes place at the initialization phase of the protocol, either by a trusted register center or by the prover. This could be done, for instance, when the author (the prover) registers his Work at an authors society (register center). The prover has first to quantize the watermark signal in order to apply a cryptographic commitment scheme working on integers. The commitment for the watermarked vector is indeed the sequence of commitments for each sample of the vector.

Homomorphic function

Usually, the detection function is a correlation or a normalised correlation between the extracted vector $\mathbf{v}$ and the watermark signal $\mathbf{w}$. The core of the protocol is the calculus of the detection function result. The prover and the verifier collaborate, but $\mathbf{w}$ shall not be disclosed.

Section 1.4.1 already introduced the use of homomorphic cryptographic functions in protocols. The correlation $\langle \mathbf{v}, \mathbf{w} \rangle$ being equal to $\sum_{i=1}^{L_X} v[i]w[i]$, we are looking for a multiplicative and/or additive homomorphism. Gopalakrishnan et al. (2001) use the

RSA public key encryption scheme (see example 16) such that $E(v[i], e)E(w[i], e) = E(v[i]w[i], e)$. Adelsbach and Sadeghi (2001) use the Fujisaki-Okamoto bit commitment scheme (which is very closed to the Okamoto-Uchiyama scheme of example 17) such that $com(\langle \mathbf{v}, \mathbf{w} \rangle) = \prod_{i=1}^{L_X} com(w[i])^{v[i]}$. In this last reference, the commitment is not a multiplicative homomorphism and the signal $\mathbf{v}$ is revealed to the prover.

Zero knowledge proof

Zero-knowledge (ZK) protocols allow a prover to demonstrate knowledge of a secret or the truth of an assertion while revealing no information to the verifier. The general structure of ZK protocols is as follows:

1. The prover sends a *commitment* to the verifier.
2. The verifier sends a *challenge* to the prover.
3. The prover makes a *response*.

Usually, the prover commits on a random value, independent of his secret. The verifier has two challenges. The first one obliges the prover to make a response depending on the random value and his secret. The random value indeed 'hides' any information concerning the secret: being uniformly drawn in the function domain, the response must be also uniformly distributed in the function range, for all secret values. The verifier has a function to verify that the response is correct with respect with the commitment. However, there is a possibility for a dishonest prover to delude this verification function with a fake commitment. This is the reason why the second challenge usually asks for the opening of the commitment. As the verifier randomly picks up one of the two challenges, the dishonest prover cannot anticipate: shall he make a correct commitment with the risk of being asked the first challenge to which he is not able to answer ignoring the secret? Shall he make a fake commitment with the risk of being asked to open it? There is one chance out of two to successfully cheat. This is the reason why the protocol lasts several rounds as the probability to delude the verifier exponentially falls. This description of ZK protocol is extremely short, the reader is invited to see proper definitions in Menezes et al. (1996).

1.4.5 Application to watermarking

We explain here two protocols whose contexts are quite different. The one proposed in Gopalakrishnan et al. (2001) is the most simple and rather tackles a buyer-seller verification proof.

Example 18 (ZK protocal from Gopalakrishnan et al. (2001)) *At the initialization, the prover gave a register center the commitment* $com(\mathbf{w})$. *The commitment function is the RSA encryption scheme with the prover's public key. Hence, anybody can use this commitment function. The prover has found a suspicious image and he has to prove the guilty seller (the verifier here) that his watermark signal is embedded in this image.*

- *The commitment. The prover generates a noise sequence* $\mathbf{z}$ *with a pseudo-random generator whose input is seed* s*, and he adds it to the extracted vector of the suspicious image:* $\mathbf{v} = \mathbf{y} + \mathbf{z}$*. He encrypts this sequence and sends* $com(\mathbf{v})$ *to the verifier.*

- *The challenge. The verifier chooses a random integer* $c = 1$ *or 2, and sends it to the prover.*

- *The response in cases where* $c = 1$*. The prover reveals* $\mathbf{v}$ *and* s*. First, the verifier checks that the sequence* $\mathbf{z}$ *given by the pseudo-random generator with seed* s *is such that* $\mathbf{v} - \mathbf{z}$ *is equal to the extracted vector* $\mathbf{y}$ *of the image in dispute. This proves that the prover has really used a pseudo-random generator (and not a fake sequence correlated with* $\mathbf{w}$*). Second, the verifier also encrypts* $\mathbf{v}$ *and compares it to the prover's commitment. This assesses that the prover is honest.*

- *The response in cases where* $c = 2$*. The prover shows that* $\mathbf{v}$ *correlates with* $\mathbf{w}$*, sending to the verifier the samples* $d[i] = v[i]w[i]$*. First, the verifier checks the homomorphic property of the commitment:* $com(d[i]) = com(v[i])com(w[i])$*,* $\forall i \in \{1, \ldots, L_X\}$*. Second, he sums the* $d[i]$*s and compares the result to a given threshold. As* $\mathbf{v} = \mathbf{w} + \mathbf{x} + \mathbf{z}$*, for large* L_X*, the correlation with* $\mathbf{w}$ *is likely to be high because* $\mathbf{z}$ *and* $\mathbf{x}$ *are statistically orthogonal to* $\mathbf{w}$*.*

The protocol works with the assumption that the pseudo-random generator cannot produce a sequence correlated with $\mathbf{w}$. At the initialization, a dishonest prover can indeed create watermark signal $\mathbf{w}$ with this generator. Hence, it is up to the trusted register center to create the watermark sequence (with a different generator). Another issue is whether or not information about $w[i]$ leaks from $d[i] = y[i]w[i] + z[i]w[i]$. Knowing $y[i]$, the value $y[i]w[i]$ discloses $w[i]$. Thus, only $z[i]w[i]$ is here to protect this knowledge. Let us assume that this random variable is distributed as $\mathcal{N}(0, \sigma_Z^2\sigma_W^2)$. Then, the information leakage is measured by $I(W; D) = 1/2 \log(1 + 1/\sigma_Z^2)$. The bigger σ_Z^2, the less information is revealed. However, this has also an impact on the correlation $\langle \mathbf{y}, \mathbf{w} \rangle$, leading to a decrease of robustness. As the protocol needs several rounds to convince the verifier, the information leakage cumulates each time challenge $c = 2$ happens, provided that sequences $\mathbf{z}$ are statistically independent: $I(W; D) = 1/2 \log(1 + n_{c=2}/\sigma_Z^2)$. Finally, although based on the general structure of cryptographic ZK protocols, this proposal is not a perfect zero-knowledge proof.

The second example is, for the moment, the most mature watermarking detection protocol. To avoid the pitfall seen above, Adelsbach and Sadeghi (2001) state that anything about the correlation must not be revealed in the clear. Even the final result of the correlation process leaks some information about the watermark sequence (see the oracle attack in section XXX). The decision being based on the comparison of the normalized correlation $\langle \mathbf{y}, \mathbf{w} \rangle / \sqrt{\langle \mathbf{y}, \mathbf{y} \rangle}$ with threshold η, an equivalent criterion is the following:

$$\begin{aligned} \langle \mathbf{y}, \mathbf{w} \rangle^2 &- \langle \mathbf{y}, \mathbf{y} \rangle \eta^2 > 0 \\ A^2 &- B > 0 \end{aligned} \tag{1.20}$$

Example 19 (ZK protocol of Adelsbach and Sadeghi (2001)) *At initialization, the prover publishes a public key and commits* $com(\mathbf{w})$ *with the Fujisaka-Okamoto commitment scheme. From the image in dispute, the verifier and the prover extract the vector* $\mathbf{y}$ *; they calculate* $com(\mathbf{y})$*, and* $B =< \mathbf{y}, \mathbf{y} > \eta^2$*. The prover sends* $com(B)$*, and he opens it. The verifier compares the opening with his own calculus of* B*. Thanks to the homomorphic propriety, the verifier calculates:*

$$com(A) = \prod_{i=1}^{L_X} com(w[i])^{y[i]} \mod n, \tag{1.21}$$

However, Eq. (1.20) needs A^2*, but the verifier is not able to make* $com(A^2)$*. This is the reason why the prover gives the verifier* $com(A^2)$*. To be sure he is not lying, there exists a cryptographic sub-protocol (Camenish and Michels (1999)) to prove the verifier that* $com(A^2)$ *really contains* A^2*. Finally, the verifier calculate the following commitment:*

$$com(A^2 - B) = \frac{com(A^2)}{com(B)} \mod n \tag{1.22}$$

Another cryptographic sub-protocol (Boudot (2000)) proves the verifier that $com(A^2 - B)$ *contains a positive number.*

This protocol is a ZK proof provided that the two sub-protocols are secure. The works of these authors is by far the most serious study about watermarking protocols (security assessments and usability). The reader is invited to go deeper in this subject with the following reference: Adelsbach and Sadeghi (2001) and Adelsbach and Sadeghi (2003).

1.4.6 Difficulties

Once again, the amount of data to be exchanged during the protocol can be very huge. Usually, a commitment is 256 bytes long. So, commitments on signal are $256L_X$ bytes long. This means several megabytes for usual vector length. Commitments also require a lot of computer power. Protocols usually last several minutes on recent PCs.

1.5 Asymmetric watermarking

Watermarking techniques were all symmetric (or private key primitives) until 1999. Symmetric means that the decoding process uses the same secret parameters than the embedder. The notion of asymmetry has been invented as a counter-measure to the possible attacks when the diversity of the secret keys and the messages to be hidden is very weak. With a classical symmetric watermarking scheme, the watermark signal tends to be the same and opponents can easily estimate it. This is especially the case of applications where a watermark detection (Is this piece of content watermarked or not?) is sufficient.

1.5.1 How to encrypt one bit?

The term 'asymmetric' is clearly misleading here as it refers to public key cryptography. We will see in subsection 1.5.4 that, in watermarking, asymmetry is not equivalent to public key. Indeed, asymmetric watermarking is more closely related to probabilistic encryption. Imagine that one has to encrypt messages with very low diversity, i.e. encrypting many times messages belonging to a very small set. Whatever the encryption scheme, it is a deterministic mapping from this set to another very small set. The number of mappings is not big enough to assure security. The opponent easily detects when the same message is sent twice, and he might gain some knowledge about the semantic meaning of each messages.

The Goldwasser-Micali or the Blum-Goldwasser algorithms for instance, are not deterministic. They are said to be probabilistic encryption schemes as they utilize randomness to cipher messages. Consequently, the same message encrypted twice gives two different cipher texts (see details in Menezes et al. (1996)).

1.5.2 Basic idea

Watermarkers have copied the concept of probabilistic encryption. The basic idea is to artificially increase the diversity of secret keys and messages to be hidden by randomizing the watermark embedding. The watermark signal $\mathbf{w}$ is a function of the secret key, the message to be hidden, but also a *random* denoted r. Consequently, two watermarked versions of same original content are different as the random changes for each embedding. The watermark signal is never the same.

The difficulty is now to build a detector which does not know the value of the random at the time the received content has been watermarked. Hence, the term asymmetric watermarking: the secret parameters at the embedding side are different from the ones at the decoding side. Indeed, the decoder looks for a given statistical property that original pieces of content do not have, which allows the detection of the watermark without knowing it. The detector is then a non-parametric test. Presented techniques are based on second order statistics testing (i.e., on autocorrelation function or spectrum).

1.5.3 Detectors based on second order statistics

References

Several asymmetric watermarking methods were invented independently. Schyndel et al. (1999) proposed an idea later on analyzed by Eggers and B. Girod (1999) and improved by J.Eggers et al. (2000). The proposal of Smith and Dodge (1999) has been re-discovered by Silvestre et al. (2001) and also by Stern (2001) with a cryptographic approach. All these methods have been proved to be conceptually the same mathematical object by Furon et al. (2001). Therefore, we only present in this short section the method from Furon and P. Duhamel (2003). It only works for watermark detection purpose. Note that very few articles address the asymmetric watermark decoding (one exception is de C.T. Gomes et al. (2000)).

Algorithms

Let us start with the embedding process. First, a white gaussian noise $\mathbf{s}$ whose power is σ_S^2 is generated thanks to the random r (for instance, r is the seed of a pseudo-random generator). Signal $\mathbf{s}$ is convoluted with a filter $h(.)$. The resulting signal $h(\mathbf{s})$ is interleaved. This interleaver acts like a pseudo-random permutation $\pi(.)$ on $\Omega = \{1, ..., L_X\}$. Its scrambles the order of the samples. We denote interleaved sequences by a tilde symbol: $\tilde{x}[i] = x[\pi(i)]$ and $x[i] = \tilde{x}[\pi^{-1}(i)]$, $\forall i \in \Omega$. Finally, the watermark signal is defined by $w[i] = h(\mathbf{s})[\pi(i)]$, $\forall i \in \Omega$. The embedding is additive with a control gain g:

$$y[i] = x[i] + g.h(\mathbf{s})[\pi(i)], \forall i \in \Omega \tag{1.23}$$

Note that signal $\mathbf{w}$ is a white gaussian noise thanks to the whitening action of the interleaver. Hence, this method can be adapted to any watermarking technique hiding white noise like spread spectrum based watermarking. Only the way to create the watermark signal and to detect its presence is different.

Normalized filter $h(.)$, random r, and permutation $\pi(.)$ are the secret parameters at the embedding side. Their operations have no impact on the signals power: $\sigma_W^2 = \sigma_S^2$. Note that any white gaussian noise is suitable and that thanks to the random, signal $\mathbf{s}$ is randomly drawn at each embedding. The detector knows the de-interleaver $\pi^{-1}(.)$ and the frequency response module of filter $h(.)$. Yet, it has no need to know signal $\mathbf{s}$. The set $\{\pi^{-1}(.), |H(\omega)|\}$ characterized the expected statistical property: The spectrum of interleaved watermarked signals is shaped like $|H(\omega)|^2$. The detection is a simple hypothesis test:

Hypothesis $\mathcal{H}_0$: Received signal $\mathbf{v}$ is not watermarked. Thanks to the supposedly ideal whitening action of de-interleaver $\pi^{-1}(.)$, the samples of $\tilde{\mathbf{v}}$ are supposed to represent a white and stationary random sequence. Hence, its spectrum $S_0(\omega)$ is flat: $S_0(\omega) = \sigma_V^2 + \mu_V^2\delta(\omega)$ where μ_V and σ_V^2 are the mean and the variance of samples $v[i]$.

Hypothesis $\mathcal{H}_1$: Received signal $\mathbf{v}$ has been watermarked. As $\tilde{\mathbf{w}}$ and $\tilde{\mathbf{x}}$ are statistically independent, their spectrum is additive: $S_1(\omega) = S_{\tilde{y}\tilde{y}}(\omega) = S_{\tilde{x}\tilde{x}}(\omega) + S_{\tilde{w}\tilde{w}}(\omega)$. As $\tilde{\mathbf{w}} = h(\mathbf{s})$, we have $S_{\tilde{w}\tilde{w}}(\omega) = g^2\sigma_S^2|H(\omega)|^2 = \sigma_W^2|H(\omega)|^2$. Finally:

$$\begin{aligned} S_1(\omega) &= \sigma_X^2 + \mu_X^2\delta(\omega) + g^2\sigma_S^2|H(\omega)|^2 \\ &= \sigma_Y^2 + \mu_Y^2\delta(\omega) + \sigma_W^2(|H(\omega)|^2 - 1) \end{aligned} \tag{1.24}$$

We used the following relationships: $\sigma_Y^2 = \sigma_X^2 + \sigma_W^2$ and $\mu_Y = \mu_X$ because $\mu_W = 0$. Note that the expected spectrum is shaped by $|H(\omega)|^2$.

Finally, the detector distinguishes signals with flat spectrum from signals with spectrum profiled as $S_1(\omega)$. The test, working on signals interleaved by $\pi^{-1}(.)$, only depends on the template $|H(\omega)|^2$. A hypothesis test of spectral analysis based on maximum likelihood is structured as follows:

$$\hat{m} = \begin{cases} 1 & \text{if } \Delta\mathcal{L} = \mathcal{L}_{L_X}(\tilde{\mathbf{v}}, S_1(\omega)) - \mathcal{L}_{L_X}(\tilde{\mathbf{v}}, S_0(\omega)) > \eta \\ 0 & \text{otherwise.} \end{cases} \tag{1.25}$$

where η is a positive threshold depending on the probability of false alarm and $\mathcal{L}_{L_X}(\tilde{\mathbf{v}}, S_i(\omega))$ is the main part of the Whittle likelihood that the spectrum of random

sequence $\tilde{\mathbf{v}}$ matches with the power spectral density $S_i(\omega)$. Its simplified expression is the following one:

$$\mathcal{L}_{L_X}(\tilde{\mathbf{v}}, S_i(\omega)) = 2L_X \int_{-\frac{1}{2}}^{\frac{1}{2}} \frac{I_{L_X}(\omega)}{S_i(\omega)} + \log S_i(\omega)\, d\omega \tag{1.26}$$

where $I_{L_X}(\omega)$ is the periodogram of signal $\tilde{\mathbf{v}}$:

$$I_{L_X}(\omega) = \left| \sum_{k=0}^{L_X-1} \tilde{v}[k].e^{2\pi i k\omega} \right|^2 \quad \forall \omega \in (-\frac{1}{2}, \frac{1}{2}]. \tag{1.27}$$

Performances

We first deal with the detection power. Usually, the requirements impose a level α of false alarm. The threshold η is chosen such that $P_{fa} < \alpha$. This leads to a Neyman-Pearson detection strategy. The test is efficient if its power (i.e., the probability to detect watermarks in watermarked content: $P_p = E[\hat{m}|\mathcal{H}_1]$) is close to one. It can be shown that the power is an increasing function of the efficiency defined by:

$$e = \frac{E[\Delta\mathcal{L}|\mathcal{H}_1] - E[\Delta\mathcal{L}|\mathcal{H}_0]}{\sigma_{\Delta\mathcal{L}|\mathcal{H}_1}}. \tag{1.28}$$

For a classical spread spectrum based technique with a detection by correlation, easy calculus give:

$$e_s = \frac{\sigma_W}{\sigma_X}\sqrt{L_X}. \tag{1.29}$$

Yet, the asymmetric methods have an smaller efficiency:

$$e_a = \frac{\sigma_W^2}{\sigma_X^2}\sqrt{L_X}. \tag{1.30}$$

As $\sigma_W/\sigma_X < 1$, asymmetric methods are less efficient than symmetric techniques. The only solution is to increase length L_X. For instance, given a power ratio σ_W^2/σ_X^2 around -20dB, one must detect watermarks with signals 10 times longer. This brings several issues about the complexity, the memory size, and the detection rate of the asymmetric detector.

The next performance criterion is the security. The role of the interleaver is very important. Of course, at the detection side, it whitens host signals which is fundamental for a good detection. But, it also plays a role from a security point of view. Ignoring the interleaver, the opponent cannot access to the domain where the core process of the detection occurs, i.e. the likelihood measurements of equation (1.26). He cannot foresee the impact of his attack.

To assess the security level of this asymmetric method, we estimate the complexity for the opponent to disclose permutation $\pi(.)$. Suppose the opponent has pairs of original and watermarked signals. The difference $\mathbf{y} - \mathbf{x}$ gives the watermark signal $\mathbf{w}$. A possibility is to try all possible permutations and to stop when interleaved signals

are colored. However, there are $L_X!$ possible permutations. For instance, with $L_X = 2048$, the Stirling formula gives the following approximation: $L_X! = 2^{19000}$ which is considerably greater than 2^{280} the estimated figure of particles in the universe!

This estimation of the security level is quite naive. The opponent knows that $\pi(.)$ is not a random permutation but generated by a pseudo-random permutation generator. This function is certainly public, and the security stems from the secrecy of the seed, a binary word whose length is $L_K << L_X$. Thus, the key space contains 2^{L_K} elements.

Furon and P. Duhamel (2003) also prove that the oracle attack is more difficult to realize with asymmetric schemes than with symmetric schemes. The main difference is that the detector is not linear. This raises the security level from $O(L_X)$ for classical correlation detector to $O(L_X^2)$ for asymmetric schemes.

Another issue is the difficulty to forge pirated content once the parameters $\{\pi^{-1}(.), |H(\omega)|\}$ have been disclosed. It is easy to modify $\tilde{\mathbf{v}}$ in order to delude the detector. It is far more difficult to predict the visual impact of the hack. This is due to the interleaver which prevents the opponent to have access in the same time to the domain where the likelihood is estimated and the domain where the perceptual impact can be measured. The opponent achieves his goal in practice after several iterations between these two domains. This shows that, in the watermarking field, asymmetry is not equivalent to public detection key technique. So far, the detection key must remain secret in the proposed asymmetric methods.

1.5.4 Some word about public detection key

The public detection key watermarking scheme is a method where the embedder has a private key and everything is disclosed (algorithm and parameters) at the detection side. Thus, a public detection key watermarking method is clearly asymmetric. One must prove that the disclosure of the detection key does not yield any security flaw.

We still restrict our analysis to watermarking detection as done by Barni et al. (2003). The hypothesis test can be considered as a map of $\mathbb{R}^{L_X}$ with two disjoint areas: the set of original signals and the set of signals regarded as watermarked. The projection attack is the following: if the opponent has disclosed the frontier between the two sets, given a watermarked content to be pirated, he now looks for the nearest signal on the other side of the frontier. In other words, this corresponds to the projection of the watermarked signal onto the frontier.

Public detection key exists if one can find an asymmetric scheme robust to the projection attack. This seems to be impossible as the knowledge of the detection algorithm and its parameters should not reveal anything about the frontier. The most promising investigation area (as far as the author knows) uses fractal frontier as done by Mansour and A. Tewfik (2002).

1.6 Conclusion

This chapter gathers four short states of the art in formerly believed 'exotic' areas of the watermarking field. Whereas relatively few works have tackled these appli-

cations (whence the ironic term 'exotic'), the trend indicates a deeper and deeper interest in authentication, fingerprinting, and protocols. This stems from the fact that the watermarking primitive cannot solve so many problems (copyright protection, copy protection, authentication...) on its own. When the application framework is considered, many issues rise and the general watermarking primitive has to be customized and inserted in a protection system. The watermarkers must have a new skill: the design of global system constituted of several security blocks (cryptographic and watermarking primitives). When the architecture of the system is completed, the watermarking primitive has a very specific role. The designer must make a security analysis (what level of robustness is required, what the aim of the pirate is, what he can do as an attack...), and then he chooses the best watermarking technology for this functionality. In a way, this chapter breaks the myth that a unique watermarking algorithm could fulfill the requirements of all applications.

Bibliography

Adelsbach A and Sadeghi A 2001 Zero-knowledge watermark detection and proof of ownership In *Information Hiding: 4th Int. Workshop* (ed. Moskowitz IS), vol. 2137, pp. 273–288 IHW 2001. Springer-Verlag, Pittsburgh, PA, USA.

Adelsbach A and Sadeghi A 2003 Advanced techniques for dispute resolving and authorship proofs on digital works *Proc. of SPIE*, pp. 677–688 Security and Watermarking of Multimedia Contents V, San Jose, Cal., USA.

Barg A and Kabatiansky G 2004 A class of I.P.P. codes with efficient identification. *Jounal of complexity* **20**(2), 137–147.

Barg A, Blakley GR and Kabatiansky GA 2003 Digital fingerprinting codes: problem statements, constructions, identification of traitors. *IEEE Trans. Inform Theory* **49**(4), 852–865.

Barni M, F. Bartolini and T. Furon 2003 A general framework for robust watermarking security. *Signal Processing* **83**(10), 2069–2084. Special issue on Security of Data Hiding Technologies, invited paper.

Barreto PS, Kim HY and Rijmen V 2002 Toward a secure public-key blockwise fragile authentication watermarking. *IEE Proc. Vision, Image and Signal Processing* **149**(2), 57–62.

Boneh D and J. Shaw 1998 Collusion-secure fingerprinting for ditial data. *IEEE Trans. on Information Theory* **44**, 1897–1905.

Boudot F 2000 Efficient proof that a commited number lies in an interval In *Proc. of Eurocrypt'00* (ed. 1807 L), pp. 431–444. Springer-Verlag, Bruges, Belgium.

Camenish J and Michels M 1999 Proving in zero-knowledge that a number is the product of two safe primes In *Proc. of Eurocrypt'99* (ed. 1592 L), pp. 107–122. Springer-Verlag, Prague, Czech Republic.

Celik M, G. Sharma, E. Saber and A. Tekalp 2002 Hierarchical watermarking for the secure image authentication with localization. *IEEE Trans. on Image Processing* **11**(6), 585–595.

Chor B and M. Naor 1994 Tracing traitors In *Proc of CRYPTO 94, Advances in cryptology* (ed. Springer-Verlag), vol. 839, pp. 257–270. Springer-Verlag.

Chor B, Fiat A, Naor M and Pinkas B 2000 Tracing traitors. *IEEE Trans. Inform. Theory* **46**, 893–910.

Coppersmith D, F. Mintzer, C. Tresser, C. W. Wu and M. M. Yeung 1999 Fragile imperceptible digital watermarking with privacy control In *Proc. of SPIE Electronic Imaging Symp., Security and Watermarking of Multimedia Contents* (ed. SPIE/IS&T), vol. 3657, pp. 79–84, San Jose, Cal., USA.

Cox I, Kilian J, Leighton T and Shamoon T 1997 Secure spread spectrum watermarking for multimedia. *IEEE Transactions on Image Processing* **6**(12), 1673–1687.

Craver S, Memon N, Yeo BL and Yeung MM 1998 Resolving rightful ownerships with invisible watermarking techniques: Limitations, attacks, and implications. *IEEE Journal on Selected Areas in Communications* **16**(4), 573–586.

de C.T. Gomes L, M. Mboup, M. Bonnet and N. Moreau 2000 Cyclostationnarity-based audio watermarking with private and public hidden data *Proc. of the 109th Convention of Audio Engineering Society*, Los Angeles, CA,USA.

Deguillaume F, Voloshynovskiy S and Pun T 2003 Secure hybrid robust watermarking resistant against tampering and copy attack. *Signal Processing* **83**(10), 2133–2170.

Eggers J and B. Girod 1999 Robustness of public key watermarking schemes V^3D^2 *Watermarking Workshop*, Erlangen, Germany.

Eggers J and B. Girod 2001 Blind watermarking applied to image authentication *Proc. of Int. Conf. on Audio, Speech and Signal Processing* IEEE, Salt-Lake City, USA.

Fernandez M and Soriano M 2004a Identification of traitors using a trellis *Proc. Information and Communications Security*, vol. 3269 of *Lecture Notes in Computer Science*. Springer-Verlag.

Fernandez M and Soriano M 2004b Soft-decoding tracing in fingerprinted multimedia content. *IEEE Multimedia* **11**(2), 38–46.

Fridrich J, M. Goljan and R. Du 2002 Lossless data embedding - new paradigm in digital watermarking. *Journal on Applied Signal Processing* **2002**(2), 185–196. Special Issue on Emerging Applications of Multimedia Data Hiding.

Friedman GL 1993 The trustworthy digital camera: Restoring credibility to the photographic image. *IEEE Trans. on Consumer Electronics* **3971**, 905–910.

Furon T and P. Duhamel 2003 An asymmetric watermarking method. *IEEE Trans. on Signal Processing* **51**(4), 981–995. Special Issue on Signal Processing for Data Hiding in Digital Media and Secure Content Delivery.

Furon T, I. Venturini and P. Duhamel 2001 Unified approach of asymmetric watermarking schemes In *Security and Watermarking of Multimedia Contents III* (ed. P.W. Wong and E. Delp). SPIE, San Jose, Cal., USA.

Gopalakrishnan K, Memon N and Vora P 2001 Protocols for watermark verification. *IEEE Multimedia* **8**(4), 66–70.

Guruswami V and Sudan M 1999 Improved decoding of reed-solomon and algebraic-geometry codes. *IEEE Trans. Inform. Theory* **45**, 1757–1767.

Holliman M and N. Memon 2000 Counterfeiting attacks on oblivious block-wise independent invisible watermarking schemes. *IEEE Trans. on Image Processing* **9**(3), 432–441.

Hollmann HDL, van Lint JH, Linnartz JP and Tolhuizen LMGM 1998 On codes with identifiable parent property. *Journal of Combinatorial Theory* **82**, 121–133.

J.Eggers, J.Su and B.Girod 2000 Public key watermarking by eigenvectors of linear transforms *Proc. of the European Signal Processing Conference* EUSIPCO, Tampere, Finland.

J.Fridrich 2002 Security of fragile authentication watermarks with localization *Proc. SPIE Photonic West*, vol. Add data for field: Volume, pp. 691–700 SPIE Electronic Imaging, San Jose, Cal., USA. Security and Watermarking of Multimedia Contents.

Kundur D and D. Hatzinakos 1999 Digital watermarking for telltale tamper proofing and authentication. *Proc of the IEEE* **87**(7), 1167–1180. Special Issue on Identification and Protection of Multimedia Information.

Kuribayashi M and Tanaka H 2003 A watermarking scheme applicable for fingerprinting protocol In *Proc. of Int. Workshop on Digital Watermarking* (ed. LNCS), pp. 532–543 IWDW'03. Springer-Verlag.

Kutter M, S. Voloshynovskiy and A. Herrigel 2000 Watermark copy attack In *Security and Watermarking of Multimedia Contents II* (ed. P.W. Wong and E. Delp), vol. 3971. SPIE Proceedings, San Jose, Cal., USA.

Lin CY and Chang S 2001 A robust image authentication method distinguishing jpeg compression from malicious manipulation. *IEEE Trans. on Circuits Syst. Video Technol.* **11**(2), 153–168.

Lu CS and H.-Y. M. Liao 2001 Multipurpose watermarking for image authentication and protection. *IEEE Trans. on Image Processing* **10**(10), 1579–1592.

Mansour M and A. Tewfik 2002 Secure detection of public watermarks with fractal decision boundaries *XI European Signal Processing Conference, EUSIPCO'02*, Toulouse, France.

Memon N and P.W. Wong 2001 A buyer-seller watermarking protocol. *IEEE Trans. on Image Processing* **10**(4), 643–649.

Menezes A, P. Van Oorschot and S. Vanstone 1996 *Handbook of applied cryptography* Discrete mathematics and its applications. CRC Press.

Pfitzmann B and Schunter M 1996 Asymmetric fingerprinting In *Advances in Cryptology* (ed. 1070 L), pp. 84–95 EUROCRYPT'96. Springer-Verlag, Berlin, Germany.

Pfitzmann B and Waidner M 1997 Anonymous fingerprinting In *Proc. of Eurocrypt'97* (ed. LNCS), pp. 88–102. Springer-Verlag, Konstanz, Germany.

Safavi-Naini R and Wang Y 2001 Collusion-secure q-ary fingerprinting for perceptual content In *Proc. Security and Privacy in Digital Rights Management, SPDRM'01* (ed. Springer-Verlag), vol. 2320 of *Lecture Notes in Computer Science*, pp. 57–75.

Safavi-Naini R and Wang Y 2003 Sequential traitor tracing. *IEEE trans. Inform. Theory* **49**(5), 1319–1326.

Schaathun HG 2004 Binary collusion-secure codes: comparison and improvements. Technical report, University of Bergen, Department of Informatics, Bergen, Norway.

Schyndel RV, A. Tirkel and I. Svalbe 1999 Key independent watermark detection *Int. Conf. on Multimedia Computing and Systems*, vol. 1, Florence, Italy.

Silverberg A, Staddon JR and Walker J 2003 Application of list decoding to tracing traitors. *IEEE Trans. Inform. Theory* **49**, 1312–1318.

Silvestre G, N. hurley, G. Hanau and W. Dowling 2001 Informed audio watermarking using digtial chaotic signals *Proc. of Int. Conf. on Acoustics, Speech and Signal Processing.* IEEE, Salt-Lake City, USA.

Smith J and Dodge C 1999 Developments in steganography In *Proc. of the third Int. Workshop on Information Hiding* (ed. Pfitzmann A), pp. 77–87. Springer Verlag, Dresden, Germany.

Staddon JR, Stinson DR and Wei R 2001 Combinatorial properties of frameproof and traceability codes. *IEEE Trans. Inform. Theory* **47**, 1042–1049.

Stern J 2001 *Contribution à la théorie de la protection de l'information* PhD thesis Université de Paris XI, Orsay Laboratoire de Recherche en Informatique.

Stinson DR and Wei R 1998 Combinatorial properties and construction of traceability schemes and frameproof codes. *SIAM Journal on Discrete Mathematics* **11**, 41–53.

Trappe W, M. Wu, Z.J. Wang and K.J.R. Liu 2003 Anti-collusion fingerprinting for multimedia. *IEEE Trans. on Signal Processing* **51**(4), 1069–1087. Special Issue on Signal Processing for Data Hiding in Digital Media and Secure Content Delivery.

Wagner N 1983 Fingerprinting *Proc. of the IEEE Symposium on Security and Privacy*, pp. 18–22, Washington, DC, USA.

Walton S 1995 Information authentication for a slippery new age. *Dr. Dobbs Journal* **4**(20), 18–26.

Wong PW 1998 A public key watermark for image verification and authentication *Proc. of Int. Conf. on Image Processing*, pp. 455–459 IEEE, Chicago, Illinois, USA.

Wong PW and N. Memon 2001 Secret and public key image watermarking schemes for images authentication and ownership verification. *IEEE Trans. on Image Processing* **10**(10), 1593–1601.

Wu CW 2002 On the design of content-based multimedia authentication system. *IEEE Trans. on Multimedia* **4**(3), 385–393.

Wu M, Trappe W, Wang ZW and Liu K 2004 Collusion-resistant fingerprinting for multimedia. *Signal Processing magazine* **21**(2), 15–27.

Wu Y 2005 Linear combination collusion attack and its application on an anti-collusion fingerprinting In *Proc. of Int. Conf. Acoustics, Speech, and Signal Processing, ICASSP'05* (ed. IEEE), vol. II, pp. 13–16, Philadelphia, USA.

Xie L and G.R. Arce 2001 A class of authentication digital watermarking for secure multimedia communication. *IEEE Trans. on Image Processing* **10**(11), 1754–1764.

Yeung M and F. Mintzer 1997 An invisible watermarking technique for image verification *Proc. of Int. Conf. on Image Processing*, pp. 680–683 IEEE, Santa Barbara, Cal, USA.

Zhao H, M. Wu, J. Wang and K.J.R. Liu 2003 Nonlinear collusion attacks on independent fingerprints for multimedia *Proc. of Int. Conf. on Acoustics, Speech and Signal Processing* IEEE ICASSP'03, Hong Kong.

Zhao Y, P. Campisi and D. Kundur 2004 Dual domain watermarking for authentication and compression of cultural heritage images. *IEEE Trans. on Image Processing* **13**(3), 430–448.

Zhu BB, M. D. Swanson and A. H. Tewfik 2004 When seeing isn't believing. *Signal Processing Magazine* **21**(2), 40–49.